Sören Bergmann

Automatische Generierung adaptiver Modelle zur Simulation von Produktionssystemen

Automatische Generierung adaptiver Modelle zur Simulation von Produktionssystemen

Sören Bergmann

Universitätsverlag Ilmenau
2014

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Wirtschaftswissenschaften der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	25. Juni 2013
1. Gutachter:	Univ.-Prof. Dr.-Ing. Steffen Straßburger (Technische Universität Ilmenau)
2. Gutachter:	Prof. Dr.-Ing. habil. Thomas Schulze (Otto-von-Guericke-Universität Magdeburg)
Tag der Verteidigung:	12. September 2013

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-86360-084-6 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2013000530

Titelfoto: photocase.com | Nortys

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Abkürzungsverzeichnis

BDE	Betriebsdatenerfassung
BPMN	Business Process Model and Notation
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CMSD	Core Manufacturing Simulation Data
DES	Diskret ereignisgesteuerte Simulation
DLZ	Durchlaufzeit
eEPK	Erweiterte Ereignisgesteuerte Prozesskette
FDD	Feature Driven Developments
GoB	Grundsätze ordnungsgemäßer Buchhaltung
GoM	Grundsätze ordnungsmäßiger Modellierung
IGES	Initial Graphics Exchange Specification
IS	Informationssysteme
KMU	Klein- und Mittelständische Unternehmen
KNN	Künstliche Neuronale Netze
MLP	Multi Layer Perceptron
OMG	Object Managment Group
PPS	Produktionsplanung und Steuerung
RELAX NG	Regular Language for XML Next Generation
SDL	Specification and Description Language
SLP	Single Layer Perceptrons
STEP	Standard for the exchange of product model data (ISO 10303)
SysML	Systems Modeling Language

UML	Unified Modeling Language
V&V	Verifikation und Validierung
WI	Wirtschaftsinformatik
XMI	XML Metadata Interchange
XML	eXtended Markup Language
XP	Extreme Programming
XSLT	eXtensible Stylesheet Language Transformation (auch XML Stylesheet Transformation)

1 Einleitung

Dieses Kapitel gibt zunächst die Motivation und Zielstellung wieder, bevor der resultierende Aufbau der Arbeit beschrieben sowie auf die angewendeten wissenschaftlichen Methoden eingegangen wird.

1.1 Motivation und Zielsetzung

In der heute vorherrschenden globalisierten Welt, in der Unternehmen mit immer kürzer werdenden Produktlebenszyklen, einer verstärkten Kundenorientierung sowie einer steigenden Individualisierung der Produkte unter hohem Kosten-, Zeit- und Qualitätsdruck agieren müssen, ist Simulation eine anerkannte Methode, um komplexe Systeme beherrschbar zu machen. Gerade die Simulation von Produktionsprozessen wird in einer Vielzahl von Branchen, unter anderem in der Automobilindustrie oder der Halbleiterfertigung von vielen Großunternehmen eingesetzt. Simulation dient hierbei zur Analyse, dem Design und der Optimierung der Produktions- und Logistikprozesse und dem dabei anfallenden Ressourceneinsatz. Simulation kann hierbei sowohl in der Planung, Inbetriebnahme als auch während des operativen Betriebs genutzt werden. Gerade in Verbindung mit dem Schlagwort Digitale Fabrik[1] sind in den letzten Jahren Simulationsmethoden verstärkt in den Fokus der Öffentlichkeit gerückt [La2007, S.669ff; BWG2009, S.1ff, We2009, S.1f].

Den unbestritten großen Potentialen der Materialflusssimulation in Unternehmen stehen entsprechend hohe Aufwände entgegen. Problematisch ist hierbei, dass der Nutzen der Simulation im Gegensatz zu den Kosten schwer zu quantifizieren ist [VDI3633, S.21f]. Dies führt gerade in klein- und mittelständischen Unternehmen (KMU) zu einem Akzeptanzproblem. Des Weiteren ist zu konstatieren, dass die Simulation in der Regel einen Experten erfordert [Ec2002, S.20]. Dieser Experte muss aufgrund der Interdisziplinarität der Simulation über detailliertes Wissen aus den Bereichen Informatik, Statistik und je nach Anwendungsfall der Betriebswirtschaftslehre sowie Ingenieurwissenschaften, z.B. Maschinenbau, verfügen [BS2010b, S.545]. Somit hängt der gesamte erreichbare Nutzen bisher stark von der Verfügbarkeit sowie den Fähigkeiten und Fertigkeiten des modellierenden Simulationsexperten ab [Be2011, S.9].

[1] "Die Digitale Fabrik ist ein Oberbegriff für ein umfassendes Netzwerk von digitalen Modellen, Methoden und Werkzeugen – u. a. der Simulation und dreidimensionalen Visualisierung –, die durch ein durchgängiges Datenmanagement integriert werden. Ihr Ziel ist die ganzheitliche Planung, Evaluierung und laufende Verbesserung aller wesentlichen Strukturen, Prozesse und Ressourcen der realen Fabrik in Verbindung mit dem Produkt" [VDI4499, S.3].

Diese ökonomischen und technischen Barrieren bei der Simulationsnutzung verhindern bzw. hemmen einen vermehrten Einsatz der Simulation in der Praxis [Jo2007, S.1673f; BWG2009, S.112]. So wird Simulation vor allem von Großunternehmen und Konzernen eingesetzt, wobei eine durchgängige Nutzung auch hier kaum zu beobachten ist. In KMUs hingegen werden die Potentiale der Simulation bisher nur in Ausnahmefällen genutzt [St2006, S.391ff; BWG2009, S.22f]. So ist einer Studie des Fraunhofer IFF zu entnehmen, dass 64% der befragten Fabrikplaner Simulation nur gezielt für einzelne spezifische Fragestellungen einsetzen, weitere 20 % der Befragten diese nie einsetzen [St2006, S.395].

Vor diesen Hintergrund wurden 2004 von Fowler und Rose [FR2004] die Reduzierung des zeitlichen Aufwandes bei der Durchführung von Simulationsstudien, die enge Integration der Simulation in die betrieblichen Abläufe sowie Interoperabilität zwischen Simulationen und betrieblichen Informationssystemen als große zukünftige Herausforderungen ("Grand Challenges") für die Simulation von Produktionssystemen definiert[2]. Das Meistern der Herausforderungen hat zum Ziel, die Verbreitung und Akzeptanz der Simulation in Unternehmen signifikant zu steigern. Viele Veröffentlichungen zeigen, dass diese oder ähnliche Anforderungen seit langem im Fokus der wissenschaftlichen Diskussionen stehen, aber bisher nicht als gelöst zu betrachten sind. So wurden in den letzten Jahren beispielsweise im Rahmen der Winter Simulation Conference, in den Panel Diskussionen "Simulation in the Future" 2000 [Ba2000] und "Panel on Grand Challenges for Modeling and Simulation" 2013 [Ta2012] ähnliche Trends und Forschungsbedarfe formuliert. So sind Forderungen wie "So Simple Anyone Can Use It" ebenso weiter hochaktuell wie die Themen Simulation on Demand bzw. webbasierte Simulation oder die Nutzung von Grid bzw. Cloud Computing im Kontext der Simulation. Des Weiteren sind Fragestellungen zu Modellierungssprachen und Interoperabilitätsstandards nach wie vor Gegenstand wissenschaftlicher Untersuchungen [Ta2012].

Untermauert werden die Aussagen durch Studien in denen vor allem Praktiker befragt wurden. Diese zeigen u.a., dass ein erheblicher Anteil des Aufwands einer Simulationsstudie auf die Datenbeschaffung und Modellerstellung entfällt [Ac1996, S.15f; MS2010, S.1ff; MS2012, S.4f]. Lösungen, die gerade diese Aufgaben unterstützen bzw. automatisieren, versprechen große Auswirkungen auf den Gesamtaufwand und die Akzeptanz. Des Weiteren ergaben Befragungen, dass gerade 6% der Fabrikplaner mit den Schnittstellen zur Simulation zufrieden sind, besonders mangelnde Anwenderfreundlichkeit (70%) und fehlende Bidirektionalität (76%) wurden bemängelt [St2006, S.395].

[2] Die Herausforderungen wurden 2002 im Rahmen eines international besetzten Workshop erarbeitet, vgl. http://www.dagstuhl.de/02351.

So stehen zum einen hohe Aufwände für die Simulation, sowohl bei der Implementierung der Modelle als auch deren Nutzung, sowie eine schlechte Integration und Standardisierung der Simulation, steigenden Anforderungen der Unternehmen bzgl. Genauigkeit, Flexibilität, Anpassbarkeit, Geschwindigkeit, Kosten, Wiederverwendbarkeit, Zyklen und phasenübergreifender Nutzbarkeit usw. gegenüber.

Ein Ansatz, der seit einigen Jahren immer wieder als ein Lösungsbeitrag für die skizzierten Probleme gehandelt wird, ist die automatische Generierung von Simulationsmodellen [Se2005, S.23; BS2010b, S.545f; Ro2009, S.588]. Unter automatischer Modellgenerierung werden verschiedene Ansätze subsumiert, die erlauben Simulationsmodelle oder zumindest Teile von Simulationsmodellen mittels Algorithmen zu erzeugen (vgl. Abschnitt 2.2). Bisher ist aber kein Ansatz veröffentlicht worden, der für einen breiteren Nutzerkreis und über einen speziellen Teilbereich hinaus gute Ergebnisse liefert. So wird die Modellgenerierung meist nur auf einen mehr oder weniger speziellen Anwendungsfall, Branche, Gewerk und Produktions-/Fertigungstyp, wie z.B. die planungsbegleitende Belieferungssimulation im Kontext der Fließfertigung einer Automobilendmontage [MS2012] zugeschnitten. In der überwiegenden Zahl der Veröffentlichungen zur Modellgenerierung werden jeweils spezielle Systeme als Datenquelle angenommen. Dies führt meist auch zur Definition und Nutzung proprietärer und oft unidirektionaler Schnittstellen. Eine Betrachtung über die reine Generierung des Modells hinaus erfolgt selten, so wird die Nutzung des Modells (Experimentplanung, Initialisierung, Ergebnisauswertung usw.) nicht bzw. kaum thematisiert. Ebenso werden in den meisten Ansätzen die dynamischen Aspekte, z.B. Reihenfolgeentscheidungen, meist nicht oder nicht ausreichend bearbeitet, auch fehlen in allen Ansätzen Methoden bzw. Vorgehensweisen, um generierte Modelle zu adaptieren, um sie an die aktuelle Wirklichkeit anzupassen und somit eine konsequente Weiternutzung bzw. Wiedernutzung zu ermöglichen [SBM2010, S.40f; BS2010b, S.574]. Schließlich ist die organisatorische Einbindung der gesamten Simulation in die betrieblichen Abläufe als Schwachpunkt zu identifizieren, da oft allein auf (software-) technische Lösungen fokussiert wird [Kl2010, S.112].

An den geschilderten Punkten setzt die hier vorliegende Dissertation an. Ziel der Arbeit ist es ein umfassendes Rahmenwerk zur Integration bzw. Automatisierung der Simulation zu entwerfen und zu validieren. Hierbei sind sowohl organisatorische, methodische als auch, zumindest prototypisch technische Komponenten zu betrachten. In diesem Zusammenhang wird die These aufgestellt, dass eine breit anwendbare automatische Modellgenerierung allein durch die Nutzung von Standards zum Datenaustausch bzw. zur Konzeptmodellerstellung sinnvoll zu implementieren ist. Weitere Anforderungen an solch einen Standard sind die Unterstützung möglichst aller Simulationsphasen, d.h. nicht allein der Modellerstellung sondern gerade auch der

Alternativenbildung, Initialisierung, Ergebnisauswertung usw. Einen geeigneten Standard zu finden bzw. zu definieren, evaluieren und dessen Anwendung für alle Phasen zu beschreiben, stellt somit die Basis der gesamten Dissertation dar. Weitere Ziele sind in der Untersuchung von Methoden zur Abbildung dynamischen Verhaltens und die Integration dieser in das Framework, sowie im Entwurf eines umrahmenden Vorgehensmodells zu sehen.

Die weitere Bearbeitung der Zielstellung erfolgt im Rahmen der Betrachtungen zum Stand der Forschung im Betrachtungsbereich (siehe Kapitel 2) und wird in Form von Forschungsfragen in Abschnitt 2.4 detailliert.

Das Vorgehen zur Erreichung der Ziele sowie der damit verbundene Aufbau der Arbeit sind dem folgenden Abschnitt zu entnehmen.

1.2 Vorgehen und Aufbau der Arbeit

Um die oben genannten Ziele zu erreichen, bedarf es eines geordneten Vorgehens und einer strukturierten Dokumentation der Ergebnisse (vgl. Abbildung 1). So werden nachdem die Motivation und Zielstellung sowie die Einordnung der Arbeit im Kontext der Wirtschaftsinformatik in Kapitel 1 thematisiert wurden, in Kapitel 2 der Stand der Forschung in den benötigten Betrachtungsbereichen wiedergegeben. Hierzu wurden entsprechende Literaturanalysen durchgeführt. Dabei wurden neben grundlegenden Lehrbüchern zum Thema Simulation, vor allem Beiträge in relevanten Zeitschriften (z.B. JOS, ZWF) und Konferenzen (u.a. der Winter Simulation Conference, ASIM Fachgruppen und Jahrestagung, PADS, European Simulation Conference, Summer Simulation Multi-Conference, Spring Simulation Multi-Conference, SimVis) sowie Dissertationsschriften im Bereich Simulation der letzten Jahre analysiert. Im Komplex Simulation von Produktions- und Logistiksystemen werden neben eher grundlegenden Definitionen benötigter Begriffe und Konzepte vor allem Simulationsvorgehensmodelle betrachtet und aufbauend die benötigten Daten untersucht. Der zweite Schwerpunkt der Literaturanalyse liegt in der Untersuchung bestehender Modellgenerierungsansätze, hierbei werden neben Schwachstellen und offenen Punkten besonders auch positive Beiträge einzelner Ansätze diskutiert. Aufbauend wird im dritten Abschnitt ein potentiell geeigneter Standard selektiert und evaluiert. Hierzu werden die ermittelten Datenbedarfe auf Abbildbarkeit hin untersucht. Der vom Autor präferierte Standard wird ausführlich vorgestellt und anderen potentiellen Standards kritisch gegenübergestellt. Abschließend werden die Erkenntnisse nochmals komprimiert zusammengefasst und die Forschungsfragen für die weitere Arbeit aufgestellt.

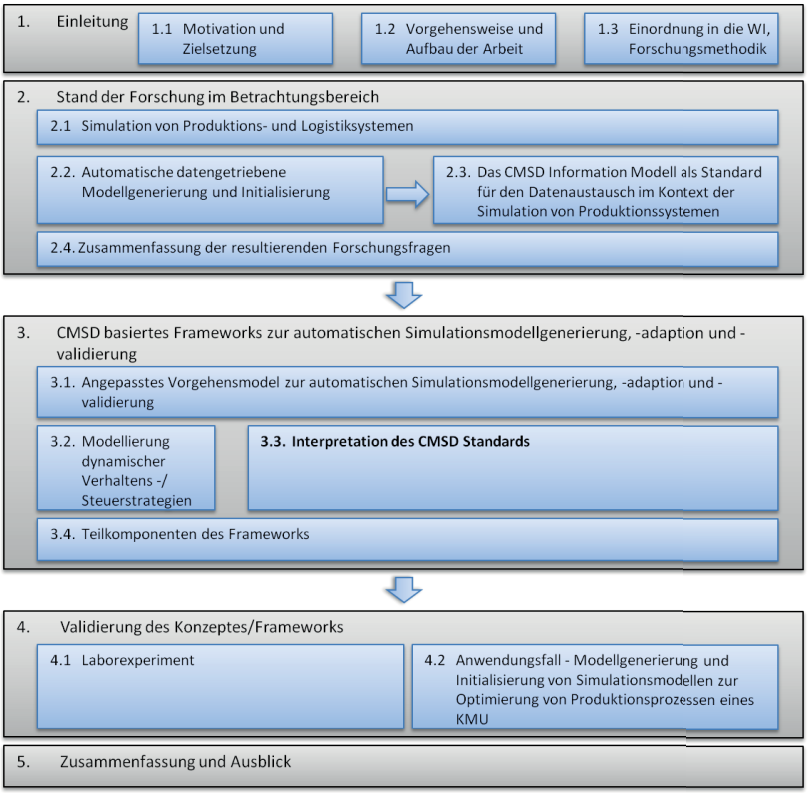


Abbildung 1: Aufbau der Arbeit

In Kapitel 3 wird der eigentliche wissenschaftliche Beitrag der Dissertation, die Konzeption und prototypische Umsetzung eines umfassenden Frameworks, vorgestellt. Zunächst wird der Rahmen des Frameworks mittels eines für die automatische Modellgenerierung mit CMSD angepassten Vorgehensmodells abgesteckt. Das Vorgehensmodell verdeutlicht die zentrale Rolle des CMSD Datenstandards für das gesamte Konzept, zudem lassen sich in folgenden Abschnitten einzelne technische Komponenten, Phasen des Modells bzw. Teilaufgaben innerhalb des Vorgehensmodells zuordnen. Mit dem Vorgehensmodell wird darüber hinaus eine bessere Integration der automatischen Modellgenerierung in die betrieblichen Abläufe angestrebt.

In Abschnitt 3.2 werden bestehende Ansätze zur Abbildung dynamischen Verhaltens aufgegriffen, analysiert und klassifiziert, sowie ein eigener auf künstlichen Neuronalen Netzen basierender Ansatz im Detail vorgestellt. Die Erkenntnisse dieses Exkurses

werden im folgenden Abschnitt, der Interpretation des CMSD Standards, vorausgesetzt bzw. direkt eingesetzt.

Die im Rahmen dieser Arbeit entstandene Interpretation des CMSD Standards wird beispielhaft anhand aufeinander aufbauender typischer Szenarien /Anwendungsfälle beschrieben. Hierzu werden jeweils alle benötigten Felder, Erweiterungen und Annahmen diskutiert. Die Ergebnisse dieses Abschnitts sind nicht allein Grundlage aller folgenden Betrachtungen, sondern auch als allgemeiner über die Grenzen dieser Dissertation hinausgehender Vorschlag im Sinne einer "Best Practice" Anwendungsempfehlung für CMSD zu sehen.

Den Abschluss des Kapitels bildet die Betrachtung der technischen Komponenten. Zunächst wird die Grobarchitektur des technischen Frameworks vorgestellt, bevor alle prototypisch implementierten Komponenten gesondert betrachtet werden. Zu den Komponenten gehören allen voran die eigentlichen CMSD basierten Modellgeneratoren, bei denen im Rahmen dieser Arbeit nochmals zwei technische Ansätze unterschieden werden können. So wird zum einen das Konzept der internen Modellgenerierung, am Beispiel des Simulators Plant Simulation, und zum anderen das Konzept der externen Modellgenerierung, am Beispiel der Simulationssprache SLX, vorgestellt, sowie kritisch bewertet. Weitere Komponenten werden benötigt um CMSD Daten auszulesen (Schnittstellen), zu validieren und ggf. anzureichern, Alternativen und Experimente zu definieren (CMSD Webfrontend) sowie Ergebnisse von CMSD basierten Experimenten auszuwerten (WebStatMonitor). Abschließend wird die Komponente zur automatischen Validierung der Simulationsmodelle thematisiert, die sowohl bei der Erstellung als auch zyklisch angewendet werden kann.

In Kapitel 4 wird das Gesamtkonzept validiert, indem alle technischen Komponenten sowie vor allem die Interpretation des Standards entsprechenden Tests unterzogen werden. Hierzu werden zum einen Labortests, d.h. Tests mit fiktiven, speziell auf bestimmte Aspekte zugeschnitten Testfällen durchgeführt, zum anderen wird das Konzept in einem KMU direkt unter Realbedingungen untersucht (Feldexperiment). Das Vorgehensmodell wird in diesem Kontext durchweg angewendet, ist aber nicht im Fokus der hier beschriebenen Validierung.

Abschließend erfolgt nochmals eine Zusammenfassung und kritische Würdigung der wesentlichen Ergebnisse. Zudem werden mögliche Erweiterungen der bestehenden Lösung diskutiert sowie ein Ausblick auf zukünftige Forschungsaufgaben gegeben.

1.3 Einordnung in die Forschung der Wirtschaftsinformatik und Forschungsmethodik

Zunächst soll die im Rahmen dieser Arbeit bearbeitete Problemstellung in die Forschung der Wirtschaftsinformatik eingeordnet werden, bevor kurz auf die genutzte Forschungsmethode eingegangen wird.

Die deutsche Wirtschaftsinformatik (WI), ebenso wie das anglo-amerikanische Pendant "Information System Research", ist eine anwendungsorientierte Wissenschaftsdisziplin, die auf betriebliche Informationssysteme[3] (IS) in Wirtschaft, Verwaltung und privatem Bereich fokussiert [WKWI2011, S.1]. Im Kontext der Wirtschaftsinformatik bzw. "Information System Research" sind sowohl gestaltungsorientierte als auch verhaltensorientierte Forschungsarbeiten zu finden, wobei in der deutschsprachigen WI der gestaltungorientierte Ansatz (engl. design science) dominiert [WH2007, S.285]. Ein Forschungsziel der WI ist u.a. "die gestaltungsorientierte Konstruktion von IS sowie die dafür notwendige (Weiter-) Entwicklung von Konzepten, Vorgehensweisen, Modellen, Methoden, Werkzeugen und (Modellierungs-) Sprachen" [Me2005, S.1f; WKWI2011, S.2].

Die Simulation von Produktionssystemen, wie in dieser Arbeit definiert, kann zweifelsfrei als IS im Sinne der WI aufgefasst werden (vgl. [WKWI2011, S.1]). Somit ist eine prinzipielle Zuordnung zum Forschungsbereich der Wirtschaftsinformatik gegeben. Konkret wird im Rahmen dieser Arbeit die Zielstellung der Weiterentwicklung von Konzepten, Vorgehensweisen, Werkzeugen und Modellierungssprachen im Kontext der automatischen Simulationsmodellgenerierung verfolgt, somit kann die Zuordnung zur gestaltungsorientierten Forschung im Sinne der WI unterstellt werden.

Idealtypisch verläuft der Prozess zum Erkenntnisgewinn iterativ in den Phasen Analyse (Exploration), Konzeption, Evaluation und Diffusion ab [Ös2010, S.667f]. Auch diese Arbeit folgt diesen Phasen.

Ziel der Analysephase ist vor allem das Ermitteln der Einflussfaktoren eines Problems. Im Rahmen der Analyse wird überwiegend die Forschungsmethode der Literaturanalyse genutzt [WKWI2011, S.4]. In der Phase der Konzeption aller Komponenten des Konzepts und auch im Rahmen der Evaluation dieser kommt eine Mischung der Forschungsmethoden konzeptionelles und argumentativ-deduktives Schließen sowie

[3] "IS sind soziotechnische Systeme, die menschliche und maschinelle Komponenten (Teilsysteme) umfassen. Sie unterstützen die Sammlung, Strukturierung, Verarbeitung, Bereitstellung, Kommunikation und Nutzung von Daten, Informationen und Wissen sowie deren Transformation. IS tragen zur Entscheidungsfindung, Koordination, Steuerung und Kontrolle von Wertschöpfungsprozessen sowie deren Automatisierung, Integration und Virtualisierung unter insbesondere ökonomischen Kriterien bei. IS können Produkt-, Prozess- und Geschäftsmodellinnovationen bewirken" [WKWI2011, S.1].

Prototyping zum Einsatz, in der Evaluation kommen Ansätze des Labor- und Feldexperiments hinzu (vgl. Tabelle 1). Die abschließende Diffusion der Ergebnisse in Wissenschaft und Praxis wird durch diese Arbeit, durch diverse Veröffentlichungen auf namhaften Konferenzen und Zeitschriften sowie das Feldexperiment sichergestellt.

Tabelle 1: Auszug aus dem für diese Arbeit relevanten Methodenspektrum der Wirtschaftsinformatik mit Anwendungsbeispielen (nach [WH2007, S.282])

Methode	Beschreibung
Formal-, konzeptionell- und argumentativ-deduktive Analyse	Logisch-deduktives Schließen kann als Forschungsmethode auf verschiedenen Formalisierungsstufen stattfinden: entweder im Rahmen mathematisch-formaler Modelle, in semi-formalen Modellen (konzeptionell, z. B. Petri-Netze) oder rein sprachlich (argumentativ, z. B. die nicht-formale Prinzipal-Agenten-Theorie). Diese drei Varianten werden als separate Methoden behandelt.
Prototyping	Es wird eine Vorabversion eines Anwendungssystems entwickelt und evaluiert. Beide Schritte können neue Erkenntnisse generieren.
Labor-/Feldexperiment	Das Experiment untersucht Kausalzusammenhänge in kontrollierter Umgebung, indem eine Experimentalvariable auf wiederholbare Weise manipuliert und die Wirkung der Manipulation gemessen wird. Der Untersuchungsgegenstand wird entweder in seiner natürlichen Umgebung (im „Feld“) oder in künstlicher Umgebung (im „Labor“) untersucht.

Für die Beschreibung und Klassifizierung der Forschungsmethoden sei hier auf entsprechende Literatur verwiesen, einen guten Einstieg bietet z.B. der Aufsatz von Wilde und Hess "Forschungsmethoden der Wirtschaftsinformatik - Eine empirische Untersuchung" [WH2007].

Abschießend ist zu bemerken, dass die Arbeit ebenso als Ausgangspunkt weiterer Forschungsarbeiten nutzbar ist.

2 Stand der Forschung im Betrachtungsbereich

Ziel dieses Kapitels ist, die grundlegenden für die weitere Arbeit relevanten Begrifflichkeiten zu definieren und den aktuellen Stand der Forschung im Betrachtungsbereich adäquat wiederzugeben.

Dazu wird die Methode der Simulation im Allgemeinen und der Simulation von Produktionssystemen im Besonderen näher betrachtet (Abschnitt 2.1). Hierbei wird zunächst grundlegend definiert, was im Folgenden unter einem Produktionssystem und unter Simulation verstanden werden soll, bevor anschließend auf Eingangsdaten und Vorgehensmodelle für Simulationsstudien sowie auf die Verifizierung und Validierung von Modellen eingegangen wird. Abschließend werden die praktischen Einsatzmöglichkeiten von Simulation im Kontext der Produktion sowie die nötige Integration der Simulation in die IT-Landschaft der Produktion andiskutiert.

In Kapitel 2.2 wird der aktuelle Stand der Forschung im Bereich Generierung und Initialisierung von Simulationsmodellen von Produktionssystemen wiedergegeben. Hierbei wird besonders darauf eingegangen, welche Forschungslücken aktuell identifiziert wurden und im Rahmen dieser Arbeit bearbeitet werden sollen.

Kapitel 2.3 widmet sich umfänglich dem Core Manufacturing Simulation Data (CMSD) Information Model, das im Rahmen dieser Arbeit als Standard für sämtlichen Datenaustausch genutzt wird. Dabei wird aufbauend auf den Erkenntnissen der vorherigen Kapiteln gezeigt, dass CMSD prinzipiell geeignet ist, um Produktionssysteme zu modellieren und welche Punkte in folgenden Kapiteln besonderer Beachtung bedürfen. Abgerundet wird dieser Abschnitt mit einer kritischen Betrachtung einer Auswahl alternativer Standards sowie einem Zwischenfazit.

Abschließend werden die Erkenntnisse dieses Abschnitts in Form von Forschungsfragen zusammengefasst.

2.1 Simulation von Produktions- und Logistiksystemen

2.1.1 Produktions- und Logistiksysteme

Wie bereits im Titel der Arbeit ersichtlich, stehen Produktionssysteme und deren Simulation im Fokus, dabei sind die Begriffe Produktionssystem bzw. Produktions- und Logistiksystem in der Literatur nicht einheitlich definiert. Aus diesem Grund wird folgend eine Arbeitsdefinition als Grundlage der weiteren Betrachtungen aufgestellt.

Unter einem System wird im Allgemeinen eine "abgegrenzte Anordnung von Komponenten, die miteinander in Beziehungen stehen" verstanden [VDI3633-1, S.3]. Unter einen Produktionssystem im Speziellen wird im Folgenden ein soziotechnisches System verstanden, welches Input (z. B. Material, Finanzmittel, Energie, Know-how,

Methoden) in wertschöpfenden (z. B. Fertigung oder Montage) und verknüpften Prozessen (z. B. Transport) zu Output (z. B. Produkte, Kosten, Reststoffe) transformiert. Hierbei ist die Aufgabe des Produktionssystems die Erstellung eines (End- oder Zwischen-) Produktes [Ev1996]. "Die Prozesse werden durch die zugeordneten technischen Ressourcen sowie den Menschen erzeugt" [NRA2008].

Solch ein Produktionssystem besteht aus Hard- und Software, wobei die Hardware die Produktionsanlagen, die Lager-, die Umschlag- und die Transporteinrichtungen umfasst. Die Regeln, die das organisatorische Zusammenspiel der Komponenten des Produktionssystems ermöglichen, werden durch die Software dargestellt [Wi2007].

Im Fokus der Betrachtungen dieser Arbeit steht die industrielle Fertigung von Stückgütern. Eine weitere Eingrenzung auf Branchen, Gewerke, Fertigungsorganisationsformen soll darüber hinaus nicht erfolgen, womit ein möglichst breiter Anwendungsbereich der Ergebnisse ermöglicht werden soll.

Des Weiteren wird der durchaus breiter fassbare Begriff Logistiksystem im Sinn der Arbeit als Teil des Produktionssystems verstanden, also im Sinne eines Intralogistiksystems. Die Intralogistik als Teilgebiet der Produktionslogistik umfasst dabei nach Arnold die Organisation, Steuerung, Durchführung und Optimierung des innerbetrieblichen Materialflusses, der Informationsströme sowie des Warenumschlages in Industrie, Handel und öffentlichen Einrichtungen [ARN2006].

2.1.2 Simulation

Simulation leitet sich vom lateinischen Begriff "simulare" (zu Deutsch nachbilden oder vortäuschen) ab, und bezeichnet eine in vielen Bereichen der Wissenschaft aber auch der Praxis weit verbreitete Methode. Simulation oder Simulierung ist hierbei "eine Vorgehensweise zur Analyse von Systemen, die für die theoretische oder formelmäßige Behandlung zu komplex sind. Dies ist überwiegend bei dynamischem Systemverhalten gegeben" [WIKI2012].

Der Verband deutscher Ingenieure (VDI) definiert in der VDI-Richtlinie 3633 Blatt 1 Simulation wie folgt:

> "Simulation ist das Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind. Insbesondere werden die Prozesse über die Zeit entwickelt.
>
> Im weiteren Sinne wird unter Simulation das Vorbereiten, Durchführen und Auswerten gezielter Experimente mit einem Simulationsmodell verstanden" [VDI3633-1, S.2].

Wie in der Definition beschrieben, nutzt Simulation immer ein Modell eines Systems (in diesen Fall Produktionssystems), dabei ist ein Modell definiert als:

> "Ein Modell ist eine vereinfachte Nachbildung eines geplanten oder existierenden Systems mit seinen Prozessen in einem anderen begrifflichen oder gegenständlichen System. Es unterscheidet sich hinsichtlich der untersuchungsrelevanten Eigenschaften nur innerhalb eines vom Untersuchungsziel abhängigen Toleranzrahmens vom Vorbild" [VDI3633-1, S.3].

Im Zuge der Nutzung der Simulation ist nie zu vernachlässigen, dass ein Modell immer eine Abstraktion der Realität darstellt, also Abweichungen des Modells von dem Realsystem per Definition von vornherein möglich sind. Diese Abweichungen können mit vertretbarem Aufwand meist nicht zu 100% verhindert werden. Bei der Modellierung ist daher gezielt darauf zu achten, dass Modellabweichungen die Ergebnisse der Simulation nicht so beeinflussen, dass keine Rückschlüsse auf das Original mehr möglich sind, d.h. Abweichung sollen nur in nicht untersuchungsrelevanten Teilaspekten oder in nicht relevanten Größenordnungen auftreten.

Um mittels Simulation zu Erkenntnissen zu gelangen, müssen Simulationsexperimente durchgeführt werden. Ein Simulationsexperiment wird wie folgt definiert:

> "Ein Simulationsexperiment ist die gezielte empirische Untersuchung des Verhaltens eines Modells durch wiederholte Simulationsläufe mit systematischer Parametervariation oder Strukturvariation.
>
> Die Simulation selbst beinhaltet keine Optimierung. Durch mathematische Optimierungsverfahren kann aber die systematische Parametervariation unterstützt und das Optimum im Hinblick auf die Simulationsziele ermittelt werden" [VDI3633-1, S.3].

Im Kontext der Produktion und Logistik hat sich für Untersuchungen zeitdynamischer Sachverhalte die diskret ereignisgesteuerte Simulation (DES)[4] durchgesetzt [We2002, S1; Ra2003, S.3; BS2010b, S.1]. Im Gegensatz zu kontinuierlicher Simulation tritt bei diskreter Simulation ein Zustandswechsel, also die Änderung eines Parameters im Simulationsmodell, nur zu bestimmten Zeitpunkten auf. Im Fall der diskret ereignisorientierten Weltsicht werden diese Zeitpunkte durch Ereignisse der Systementitäten/Modellobjekte determiniert. Eine weitere Weltsicht der diskreten

[4] Neben dem hier verwendeten Begriff der "diskret ereignisgesteuerten Simulation" (wie z.B. in [Ho2007]), sind in der Literatur auch die Begriffe "Ereignisdiskrete Simulation" (z.B. in [RSW2008]), "diskret ereignisorientierte Simulation" (z.B. in [MS2012]) oder im anglo-amerikanischen Raum "Discrete Event Simulation" (z.B. in [La2007]) gebräuchlich.

Simulation ist die prozessorientierte Weltsicht, bei der nicht Ereignisse der Objekte per se, sondern der Prozess des Durchlaufs einzelner Objekte (z.B. Kunden, Werkstücke, Aufträge) durch das System modelliert werden. Des Weiteren können Simulationsmodelle anhand einer Vielzahl von zusätzlichen Merkmalen klassifiziert werden, an dieser Stelle sei nur noch das Merkmal des Zufallseinflusses explizit aufgeführt. Der Zufallseinfluss gibt an, ob einzelne Parameter stochastischen Einflüssen unterliegen (stochastisches Simulationsmodell) oder ob alle modellierten Parameter unter gleichen Bedingungen immer auch gleiche Werte annehmen (deterministisches Simulationsmodell) [KG1995, S. 17-22]. Im Rahmen dieser Arbeit wird im Folgenden auf die diskret ereignisorientiert bzw. prozessorientierte Weltsicht mit stochastischen Einflüssen fokussiert.

2.1.3 Anwendungen der Simulation im Kontext der Produktion

Diskret ereignisgesteuerte Simulation findet im Kontext der Produkt bereits seit vielen Jahre diverse Anwendungsmöglichkeiten, so wird durch den VDI in der Norm 3633 Blatt 1 bereits konstatiert, dass "der Schwerpunkt des Simulationseinsatzes [..] ursprünglich auf der Planungsabsicherung" [VDI3633-1, S.2] lag. Zu beobachten ist aber, dass in den letzten Jahren die diskret ereignisgesteuerte Simulation auch in weiteren Phasen des Planungs- und Realisierungsprozesses über die Phase der Inbetriebnahme bis hin zum operativen Betrieb Verwendung findet [VDI3633-1, S.2ff]. Eine abschließende Liste von Anwendungsklassen lässt sich in diesen dynamischen Umgebungen schwer erstellen, einige Szenarien sollen aber an dieser Stelle erwähnt werden, so werden von Müller-Sommer typische Einsatzfälle im Umfeld der automobilen Logistik identifiziert:

- Werkssimulation
- Belieferungssimulation
- Supply-Chain-Simulation
- Verkehrsflusssimulation [MS2012, S. 30f].

Besondere Aufmerksamkeit findet Simulation (im Speziellen die DES) in den aktuellen Publikationen zum Thema Digitale Fabrik. Sie definiert sich wie folgt:

> "Die Digitale Fabrik ist der Oberbegriff für ein umfassendes Netzwerk von digitalen Modellen, Methoden und Werkzeugen – u. a. der Simulation und der drei dimensionalen Visualisierung – , die durch ein durchgängiges Datenmanagement integriert werden.
>
> Ihr Ziel ist die ganzheitliche Planung, Evaluierung und laufende Verbesserung aller wesentlichen Strukturen, Prozesse und Ressourcen der realen Fabrik in Verbindung mit dem Produkt" [VDI4499, S.3].

Neben diesen typischen planungsbegleiten Anwendungsfällen werden in diversen Publikationen, z.B. Systeme:

- zur simulationsbasierten Fertigungssteuerung, z.B. in [Sc2002]
- für simulationsbasierte Frühwarnsysteme, z.B. in [Ho2007]
- zur virtuellen Inbetriebnahme auf Prozessleitebene/Emulation, z.B. in [MS2012]
- zur simulationsbasierten Entscheidungsunterstützung, z.B. in [SBS2012]

vorgestellt.

So vielfältig die Anwendungsfälle erscheinen, so liegt doch allen die gleiche Methodik der Simulation zu Grunde, wobei natürlich in Bezug auf Detaillierung, Granularität aber auch bezüglich verfügbarer Daten und deren Qualität, Anforderungen an die Darstellung und Performance, der Einbindung in die betriebliche IT, Ergebnisdaten und einiger weiterer Faktoren durchaus erhebliche Unterschiede existieren können.

2.1.4 Grundsätze der ordnungsgemäßen Modellierung

In vielen wissenschaftlichen Disziplinen, aber gerade auch in der Wirtschaftsinformatik, werden betriebliche Abläufe und Strukturen auf unterschiedlichste Art und mit verschiedenstem Fokus modelliert. Wie in den vorangegangenen Abschnitten dargelegt, können beispielsweise Simulationsprojekte eine Modellierung nötig machen. Dabei ist unabhängig vom Simulationskontext sicherzustellen, dass die erzeugten Modelle neben reiner syntaktischer Korrektheit Mindestanforderungen hinsichtlich der Qualität, Klarheit und Konsistenzsicherung erfüllen, um die Einsatzfähigkeit (je Modellzweck ggf. sehr unterschiedlich) zu gewährleisten. Dazu wurden durch Becker, Rosemann und Schütte 1995, analog zu den aus dem Rechnungswesen bekannten Grundsätzen ordnungsgemäßer Buchhaltung (GoB), die Grundsätze ordnungsmäßiger Modellierung (GoM) definiert [BRS1995, Be1998].

Grundlegend wurden folgende 6 Grundsätze beschrieben [vgl.BRS1995 S 437-439; Be98 S. 4-7]:

- **Grundsatz der Richtigkeit**: In seinen wesentlichen Teilen muss das Modell der modellierten Realität entsprechen.
- **Grundsatz der Relevanz**: Es sollten nur die Aspekte modelliert werden, die für den Modellzweck des realen Systems relevant sind. So hat beispielsweise die Modellierung zum Zweck der Simulation im Planungskontext eine andere Relevanz als zum Zweck der simulationsbasierten Fertigungssteuerung oder der Geschäftsprozessmodellierung zu Zertifizierungszwecken. Dabei muss auf den entsprechenden Detailierungsgrad geachtet werden.

- **Grundsatz der Wirtschaftlichkeit:** Theoretisch kann gesagt werden, dass die Wirtschaftlichkeit dann gegeben ist, wenn der Grenznutzen gleich den Grenzkosten der Modellbildung ist, d.h. es ist die Frage zu beantworten, ob die Zusatzkosten einer detaillierteren oder umfänglicheren Modellierung durch die zu erwartenden verbesserten Ergebnisse aufgewogen werden.
- **Grundsatz der Klarheit:** Modelle sollten so detailliert wie nötig und so einfach wie möglich gehalten werden, dabei steht Leserlichkeit, Verständlichkeit und Anschaulichkeit im Vordergrund.
- **Grundsatz der Vergleichbarkeit:** Anforderung, dass mit unterschiedlichen Modellierungsverfahren (z.B. EPK und Petrinetze) erstellte Modelle miteinander vergleichbar sein sollen.
- **Grundsatz des systematischen Aufbaus:** Die Konsistenz aller (Teil-)Modelle untereinander muss gesichert sein, so muss eine modell- bzw. sichtübergreifende referentielle Integrität gewährleistet werden.

Die vorgestellten GoM sind allerdings nicht mit konkreten Handlungsanweisungen (z.B. Namenskonventionen, Ordnungsrahmen usw.) hinterlegt, vielmehr fließen sie in die im folgenden Abschnitt beleuchteten Vorgehensmodelle für Simulationsstudien und Validierungsverfahren von Simulationsmodellen ein. Bei der Diskussion der automatischen Modellgenerierung und der Ausgestaltung des Frameworks im weiteren Verlauf dieser Arbeit wird entsprechend auf den Einfluss einzelner Aspekte bzgl. der Einhaltung/Verbesserung der GoM eingegangen.

2.1.5 Vorgehensmodelle für Simulationsstudien

Vorgehensmodelle sind eine in verschiedensten Bereichen verbreitete Methode um hohe Komplexitäten beherrschbar zu machen. Dabei werden üblicherweise verschiedene Phasen bzw. Blöcke definiert, die jeweils mit einem Satz an Verfahrensweisen und Methoden hinterlegt werden [WIKI2012a, Br2010]. Ziel der Vorgehensmodelle ist es allgemein, einen geordneten Entwicklungsprozess zu sichern. Darüber hinaus wird zur Einhaltung der GoM (vgl. Abschnitt 2.1.4) beigetragen, auch wenn dies nicht explizit angeführt wird oder als Ziel der Vorgehensmodelle definiert ist.

Üblicherweise werden in Vorgehensmodellen verschiedene Phasen definiert, sowie geeignete Methoden je Phase vorgeschlagen. In der Vergangenheit wurden auch für die Simulation eine Reihe von Vorgehensmodellen mit unterschiedlichem Fokus propagiert. Viele der Vorgehensmodelle ähneln sich dabei in Teilen, wobei Benamungen, Komplexität und Umfang durchaus unterschiedlich sein können.

Folgende Komponenten sind durchweg in den meisten Vorgehensmodellen, wenn auch teils zusammengefasst oder unter anderen Namen, zu finden:

- Aufgabenanalyse
- Modellformulierung
- Modellimplementierung
- Modellüberprüfung
- Modellanwendung [RSW2008, S. 29].

Statt der Modellüberprüfung werden in den meisten Quellen üblicherweise die Begriffe Verifikation und Validierung aufgeführt, dabei wird meist aber nicht allein auf die Überprüfung des Endprodukts Simulationsmodell sondern auf alle Zwischen- und Teilergebnisse abgezielt. Verifikation ist hierbei mit der Fragestellung „Ist das Modell richtig?“ (Are we creating the X right?) und die Validierung mit „Ist es das richtige Modell?“ (Are we creating the right X?) verknüpft [Ba2003]. Aufgrund der Komplexität ist eine formale Prüfung meist nicht oder schwer möglich, zumal der Abstraktionscharakteristika von Modellen folgend nur eine auf den Modellzweck und das Abstraktionsniveau angepasste hinreichende Genauigkeit angestrebt werden sollte [RSW2008, S14f]. In diesem Sinne ist eigentlich die Glaubwürdigkeit (Credibility) des Modells zu überprüfen vgl. [Ca2002].

Im Folgenden sollen kurz einige in der Praxis und wissenschaftlichen Literatur weit verbreitete Vorgehensmodelle vorgestellt werden. Jedes dieser Modelle trägt eigene Aspekte für das in Abschnitt 3.1 vorgestellte Vorgehensmodell zur automatischen Modellgenerierung, -initialisierung und -validierung bei. Auf detaillierte Beschreibungen jedes Bestandteils aller Vorgehensmodelle wird verzichtet und auf die entsprechend angegebene Literatur verwiesen. Ziel des Abschnittes ist es, mögliche Schwächen für den Anwendungsfall automatische Modellgenerierung anzudeuten, aber auch im angepassten Vorgehensmodell aufgegriffene Konzepte einzuordnen.

Bereits 1982 beschrieb Robert G. Sargent in einen Aufsatz mit dem Titel "Verification and Validation of Simulation Models" [Sa1982] ein Vorgehensmodell mit geringer Komplexität. Anzumerken ist, dass, wie im Titel des Aufsatzes zu entnehmen ist, vor allem auf die Verifikation und Validierung (V&V) im Prozess Modellierung fokussiert wird, die eigentliche Modellanwendung, d.h. Experimentdefinition, -durchführung, -dokumentation, -auswertung und die Anwendung der Erkenntnisse sind nicht bzw. kaum Bestandteil dieser Version des Vorgehensmodells.

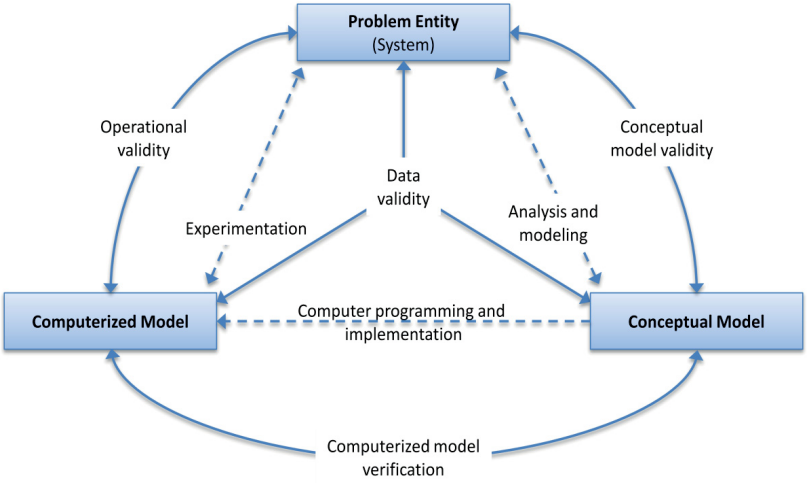


Abbildung 2: Vorgehensmodell nach Sargent; Simplified version of the modeling process [Sa1982, Sa 2011, S. 186]

Neben den V&V Aktivitäten (vgl. Abbildung 2, Volllinien Pfeile) ist eine Trennung zwischen betrachtetem System, Konzeptmodell [5] und dem implementierten Simulationsmodell ein für weitere Betrachtungen wesentlicher Punkt in den Arbeiten von Sargent [Sa1982; Sa2011]. Hierbei ist der Prozess der Konzeptmodellerstellung (Analysis and modeling) sowie der Simulationsmodellimplementierung (Computer programming and implementation) iterativ, d.h. er wird ggf. solange durchgeführt, bis ein ausreichend valides bzw. verifiziertes Modell erzeugt wurde, bevor die nächste Phase beginnen darf. Bezüglich der Experimentphase werden keine expliziten Angaben bzgl. möglicher Wiederholungen gemacht, wobei angenommen werden kann, dass Experimente zur statistischen Absicherung und/oder bei Vorliegen mehrerer Szenarien wiederholt werden [Sa2010, S. 168-171]. Eine separate Betrachtung der Datenerhebung, der Experimentplanung bzw. Alternativenentwicklung und der Ergebnisauswertung und -umsetzung erfolgt kaum, ebenso ist keine Anpassung an den Einsatz von z.B. Bausteinen, automatischer Modellgenerierung, der betriebsbegleitenden, d.h. kontinuierlichen Simulationsnutzung usw. vorgesehen.

[5] Das Konzeptmodell wird nach Robinson definiert als: „The conceptual model is a non-software- specific description of the simulation model that is to be developed, describing the objectives, inputs, outputs, content, assumptions and simplifications of the model" [Ro2004, S.65]. Prinzipiell sind formale als auch informale Konzeptmodelle denkbar, in der Praxis sind formale Beschreibungen aber kaum anzutreffen.

Das zweite hier näher betrachtete Vorgehensmodell für Simulationsstudien (vgl. Abbildung 3) wurde von Averill M. Law in den letzen Jahrzehnten in verschiedenen Veröffentlichungen publiziert, unter anderen in [La2007, S.67], [LC1991, S.22] und [La2008, S.41].

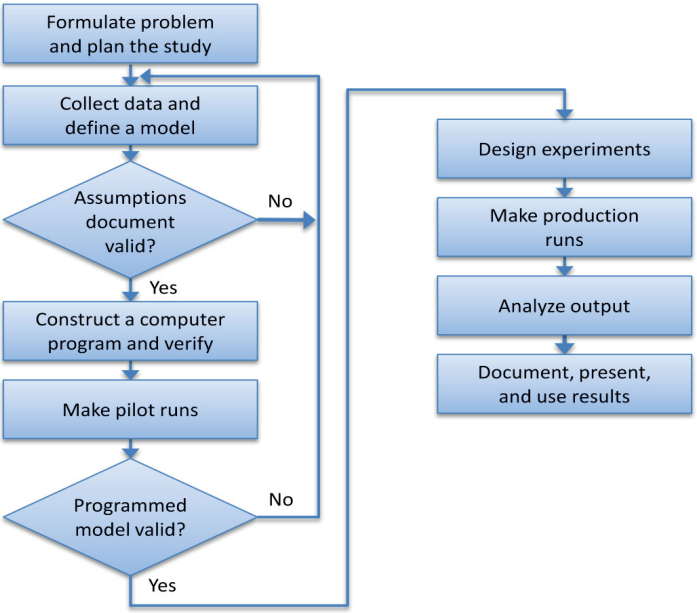


Abbildung 3: Vorgehensmodell nach Law; Steps in a simulations study [La2007, S.67]

Neben den schon bei Sargent erkennbaren V&V Aktivitäten und der wenn auch nicht so explizit formalisierten Trennung in Konzeptmodell (Collect data and define model) und Simulationsmodellerstellung (Construct a computer program and verify) ist bei Law besonders die zeitliche und logische (ggf. personelle) Abgrenzung der Modellerzeugung, d.h. Modellgenerierung, und des Experimentdesigns und -durchführung, inklusive der hier nötigen Aufgaben, wie z.B. der Initialisierung des Models und der Erzeugung entsprechender Outputdaten (vgl. Abschnitt 2.1.6 und 2.2) zu erkennen.

Bezüglich der Anpassung an spezielle Anwendungsfälle gelten ähnliche Einschränkungen wie bei Sargent. So ist auch dieses Vorgehensmodell unter der Annahme zu sehen, dass Simulation im Rahmen eines mehr oder weniger abgeschlossenen Projektes zum Einsatz kommt, ein Weiternutzen oder zyklisches Einsetzen des Modells, z.B. im operativen Betrieb zur Steuerung, ist nicht vorgesehen. Auch Reaktionen zur Anpassungen

(Adaption) des Modells an während der Projektzeit auftretende Änderungen / Konkretisierungen des zu modellierende Realsystems sind nicht Teil des Vorgehens.

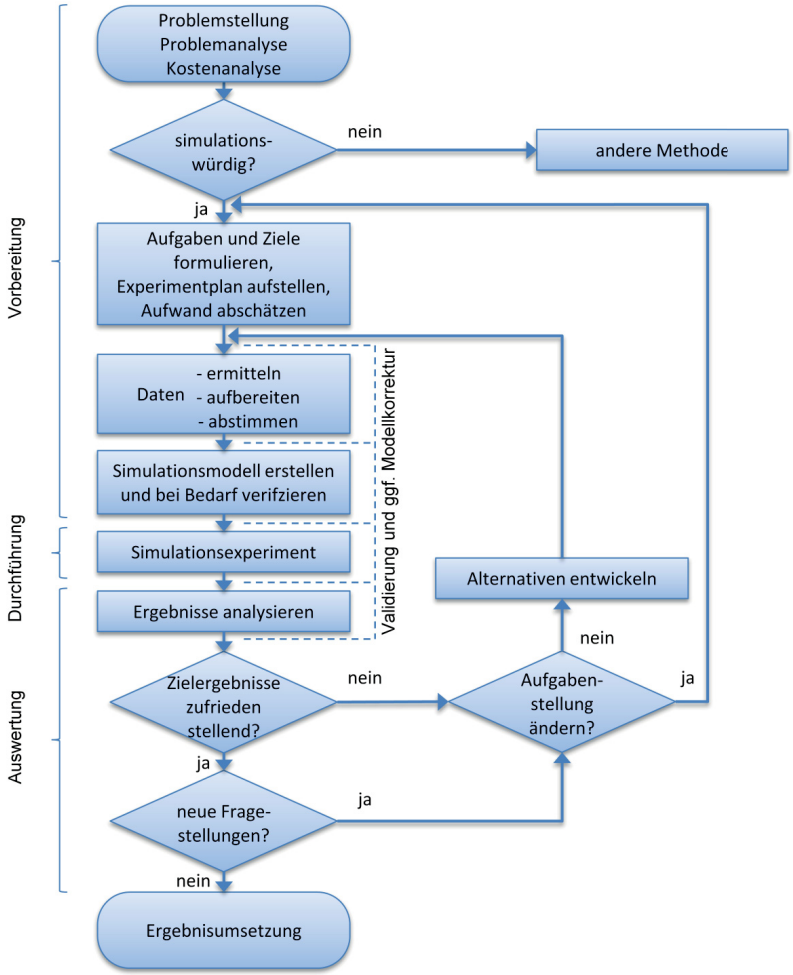


Abbildung 4: VDI Vorgehensweise bei einer Simulationsstudie [VDI3633, S.11]

Das im deutschsprachigen Raum wohl am häufigsten genutzte Vorgehensmodell und das, der hier vorgestellten, einzige standardisierte ist die Vorgehensweise bei einer Simulationsstudie (vgl. Abbildung 4), die Teil des VDI Standards 3633 - Blatt1 "Simulation von Logistik-, Materialfluß- und Produktionssystemen - Grundlagen" ist

[VDI3633, S. 11]. Weiter nennenswert ist das auf dem VDI Modell basierende Vorgehensmodell von Raabe, Spieckermann und Wenzel [RSW2008, S.5f], in welchen vor allem Aspekte der Parallelität von Datenerhebung und -aufbereitung zur Systemanalyse, Konzeptmodellerstellung und Simulationsmodelimplementierung sowie zur Verifikation und Validierung der Daten und Modelle deutlich werden.

Wie in folgenden Abschnitten noch weiter ausgeführt, ist in beiden Modellen die fehlende Trennung zwischen Konzeptmodell und Simulationsmodell und zwischen Modellaufbau und Nutzung/Experiment im Kontext der automatischen Modellgenerierung problematisch. Durchaus positiv, wenn auch nicht abschließend sind die vorgesehenen Iterationen, nicht nur im Zuge der V&V sondern vor allem auch der wiederholten Nutzung im Sinne von Alternativen, Designfehlern und geänderten Rahmenbedingungen.

Zusammenfassend kann gesagt werden, dass keines der Vorgehensmodelle die automatische Modellgenerierung geeignet unterstützt bzw. überhaupt betrachtet. Schwachpunkte aller Vorgehensmodelle für die in dieser Arbeit angestrebte automatische Modellgenerierung, -initiierung, -adaption und Validierung sind vor allem eine starke Fokussierung auf den Projektcharakter (keine/kaum Wiedernutzung, kaum Iteration, kein Änderungsmanagement), keine Trennung unterschiedlicher Rollen (z.B. Simulationsexperte, Softwareentwickler, Fachbereichsmitarbeiter), keine konkrete Einbeziehung von (Software-)Schnittstellen zu Datenquellen, keine Betrachtung von automatisch generierten Modellen, keine Betrachtung von Spezifika der Nutzung und Entwicklungen von Bausteinen usw.

Neben den vorgestellten Vorgehensmodellen für Simulationsstudien werden in der Literatur separat Vorgehensmodelle für die V&V im Kontext der Simulation aufgeführt, beispielhaft sei hier nur das Vorgehensmodell zur V&V für die Simulation in Produktion und Logistik von Rabe, Spieckermann und Wenzel [RSW2008, S.118ff] genannt. In diesem Modell werden einzelnen Phasen einer Simulationsstudie Phasenergebnisse getrennt nach Modell und Daten zugeordnet, diese Ergebnisse sind dann entweder das zu prüfende Objekt und/oder dienen als Bezugsobjekt gegen das getestet wird. Zusätzlich werden meist passend V&V Techniken aus einem Katalog möglicher Techniken vorgeschlagen [RSW2008, S. 93-113].

Hervorzuheben ist, dass die Autoren ein Tailoring (maßschneidern), also eine Anpassung des Umfangs der Testmethoden an bestimmte Rahmenbedingungen koppeln. So werden unter anderem für bausteinorientierte Simulationswerkzeuge, automatische Modellgenerierung und betriebsbegleitende Simulation Denkanstöße für ein sinnvolles Tailoring gegeben [RSW2008, S130-134].

2.1.6 Eingangsdaten der Simulation

Wie in allen Vorgehensmodellen gleichermaßen erkennbar, spielen Daten in Simulationsstudien eine entscheidende Rolle. Dabei kann eine Vielzahl von Daten relevant sein. Menge, Art und Qualitätsanforderungen können dabei stark vom Ziel, dem zu simulierenden System und vielen weiteren projektspezifischen Parametern abhängen. Nach der VDI Norm 3633 Blatt 1 [VDI3633-1, S,12f] lässt sich die Simulationsdatenbasis grob in 3 Klassen einteilen, siehe Abbildung 5.

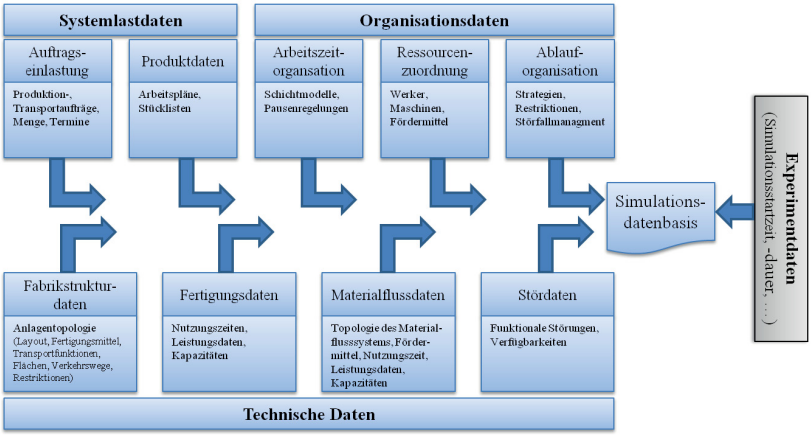


Abbildung 5: Eingangsdaten der Simulation (in Anlehnung [VDI3633-1, S.13]

Die Klasse der technischen Daten beschreibt das Layout und die Topologie des gesamten Systems sowie die Eigenschaften der einzelnen Systemkomponenten. Die Organisationsdaten spezifizieren die Ablauf- und Prozessorganisation, vor allem Arbeitszeitmodelle, Ressourcenzuordnungen und -Strategien. Schließlich beschreibt die Klasse der Systemlastdaten vor allem Arbeitsaufträge und deren Eigenschaften. Während die technischen und organisatorischen Daten eher als statisch zu sehen sind und meist für die Modellerstellung direkt benötigt werden, werden die Systemlastdaten als veränderlich angesehen und werden vor allem in der Experimentphase, d.h. bei der Ausführung der Simulation benötigt. Ein spezieller Fall innerhalb der Eingangsdaten stellen Daten bzgl. der inneren Modellogik bzw. des Modellverhaltens, wie z.B. Steuerungslogiken, Reihenfolgeregeln usw. dar. Diese sind in die Klasse der Ablauforganisationsdaten, im Speziellen den Strategien und Restriktionen, einzuordnen. Eine weiterführende Betrachtung dieser Daten erfolgt in Abschnitt 3.2 [BSS2011a, S. 2226; BSS2011b, S. 4ff].

2.1.7 Simulationsergebnisanalyse - Statistik und Kennzahlen

Neben den Eingangsdaten ist eine zweite Gruppe von Daten im Kontext der Simulation von besonderer Bedeutung, die Ergebnisdaten, die bei der Durchführung einzelner Simulationsläufe bzw. kompletter Experimente anfallen. Unter einem Experiment wird in diesem Sinne die „[…] gezielte empirischen Untersuchung des Modellverhaltens über einen bestimmten Zeithorizont durch wiederholte Simulationsläufe mit systematischer Parametervariation“ verstanden [VDI3633-3, S. 2].

Es können grob zwei Ziele der Ergebnisdatenanalyse (engl. "Output Analysis") unterschieden werden, zum einen kann die absolute Leistungsfähigkeit einer bestimmten Parameterkombination ermittelt werden, z.B. mittels simulativer Ermittlung der zu erwartenden Ausbringungsmenge, zum anderen können verschiedene Konfigurationen verglichen und in einen relativen Zusammenhang gebracht werden [LK2000, S. 86]. Damit eng verbunden ist eine sinnvolle Bestimmung des Experimentplans, d.h. der Festlegung von Parametervariationen, Anzahl der Replikationen usw., detaillierte Betrachtungen zur Experimentplanung sind bspw. in [VDI3633-3, S. 5-8], [Ba2010] oder [LK2000, S. 622–625] zu finden.

Grundlegend kann gesagt werden, dass "das Ergebnis eines Simulationslaufes [...] aus Datensätzen, die Auskunft geben über die zeitabhängigen Zustandsänderungen der ortsfesten und dynamischen Modellelemente" besteht [VDI3633-3, S.9]. Diese sog. Trace Daten werden üblicherweise an festgelegten Messpunkten gespeichert und sind mit den Daten der Betriebsdatenerfassung [6](BDE) einer realen Produktion vergleichbar. Die Rohdaten sind meist erst nach entsprechender Aufbereitung handhabbar und bilden die Grundlage für weitere statistische Auswertungen. Das Aufbereiten kann hierbei während und/oder nach dem eigentlichen Simulationslauf bzw. gesamten Experiment erfolgen. Typischerweise kann hierbei zwischen der auftragsorientierten und der elementorientierten Sichtweise (vgl. Tabelle 2) unterschieden werden [VDI3633-3, S.9].

Tabelle 2: Elemente der Ergebnisdaten nach VDI-Richtlinie 3633 Blatt 3 [VDI 3633-3, S. 9]

Sichtweise	Elemente	Beschreibung
Auftrags-orientiert	Daten der Aufträge	Hauptselektions- und Hauptsortierungskriterium bilden die Aufträge, welche durch das System laufen. So sind Informationen bzgl. Auftragsnummer, geplanter bzw. realisierter Termine der Freigabe sowie Fälligkeit und bestimmten Arbeitsvorgängen von Interesse.

[6] "Die Betriebsdatenerfassung (BDE) dient der Erfassung und Ausgabe betrieblicher Daten in maschinenverarbeitbarer Form. […] BDE ist eine notwendige Voraussetzung für eine Automatisierung der Fertigungssteuerung" [Mö2013].

Element-orientiert	Daten mobiler Elemente	Diese Elemente bewegen sich zum Transport von Objekten (z. B. Werkstücke, Komponenten) durch das Modell. Hier sind vor allem die Transportzeiten wichtig.
	Daten stationärer Elemente	Objekte bewegen sich während des Simulationslaufs durch das Modell und treten hierbei in stationäre Elemente ein und verweilen dort bis zu ihrem Austritt aus dem Element. Für stationäre Elemente sind bspw. die Anzahl der Eintritte, die Kapazität sowie der Nutzungsgrad interessant.
	Daten statistischer Elemente	Statistische Elemente existieren nur logisch und bilden Verteilungen von Zustandsvariablen (z.B. für Verfügbarkeiten) an Schnittstellen von Bausteinen, die als sogenannte „black box“ betrachtet werden, ab.
	Daten von Warteschlangen	Hierbei handelt es sich bspw. um Daten zum zeitabhängigen Inhalt der Warteschlangen oder die Verweildauer einzelner Elemente.

Die Abbildung der Rohdaten im implementierten Framework wird in Abschnitt 3.3.9 nochmals detailliert.

Im Rahmen der Datenaufbereitung werden die Rohdaten zu Kennzahlen verdichtet, womit die Aussagekraft der in ihnen enthaltenen Informationen erhöht wird, ebenfalls erfolgt meist eine Selektion von bestimmten Daten, eine Sortierung sowie ggf. eine Umrechnung der Rohdaten. Eine weitere essentielle Teilaufgabe der Datenaufbereitung stellt die Datendarstellung inklusive der Visualisierung der Daten in entsprechenden Diagrammen dar, da dies in den meisten Fällen die nachgelagerte Interpretation und Variantenbewertung erheblich erleichtert bzw. überhaupt erst ermöglicht. Der Schritt der Interpretation hat zum Ziel, die Ergebnisdaten mit Einflussgrößen in Beziehung zu setzen [VDI 3633-3, S. 3]. Dabei sind aber für eine korrekte Interpretation der Ergebnisse die Einbeziehung der im Zuge der Simulationsstudie getätigten Annahmen, z.B. bzgl. der Eingangsdaten, zwingend erforderlich [RSW2008, S. 82].

Wie bereits beschrieben fokussiert diese Arbeit auf DES mit stochastischen Einflüssen. In den meisten solcher Modelle wirkt sich bereits ein nicht deterministisches Element durch die Wechselwirkungen im Modell auf das Gesamtergebnis aus. Um eine Fehlinterpretation zu vermeiden und zuverlässige Aussagen über das Modell treffen zu können, ist eine statistische Auswertung unter Verwendung von Replikationen notwendig [LK2000, S. 247]. Grundlage der diskreten statistischen Auswertung stellen statistische Kennwerte dar (vgl. Tabelle 3), diese können sowohl für einen Lauf gebildet werden, z.B. durchschnittliche Durchlaufzeit aller Aufträge eines Simulationslauf, oder über verschiedene Läufe, z.B. Durchschnittswert der durchschnittlichen Durchlaufzeit über alle Replikationsläufe.

Basierend auf den empirisch belegten statistischen Maßzahlen arithmetisches Mittel, empirische Varianz und empirische Standardabweichung können der Erwartungswert, die Varianz und die Standardabweichungen geschätzt werden. Die Schätzung ist nötig, da der wahre Wert nicht exakt aus dem empirisch ermittelten hervorgeht. Die Simulationsergebnisse können als Punktschätzung genutzt werden, wodurch unter Einbeziehung der Werte[7] n, X, x_i und p_i die Kennwerte abgeleitet werden können [BaVo08, S. 27, 32, 109; La2007, S.485-533; Kü06, S. 283–286]:

Tabelle 3: Auszug grundlegender statistischer Kennwerte (in Anlehnung an [BV2008, S. 27, 32, 109], [Ku2006, S. 283–286])

Statistische Maßzahl	Berechnungsvorschrift für den Lageparameter	Zu schätzender Parameter
Arithmetischer Mittelwert	$\overline{x} = 1/n \sum_{i=1}^{n} x_i$	Erwartungswert: $E(X) = ì$
Modalwert	Häufigster angenommener Wert	
Empirische Varianz	$s^2 = 1/(n-1) \sum_{i=1}^{n} (x_i - \overline{x})^2$	Varianz: $Var(X) = ó^2$
Empirische Standardabweichung	$s = \sqrt{s^2}$	Standardabweichung: $ó = \sqrt{Var(X)}$
Minimum	$\min x_i$	
Maximum	$\max x_i$	

Weitere statistische Maßzahlen, die typischerweise im Kontext der Datenauswertung zum Einsatz kommen, sind z.B. der empirische Median [8], das untere ($x_{0,25}$) und obere Quartil [9] ($x_{0,75}$) sowie Konfidenzintervalle [10]. In allen Auswertungen ist ggf. der Zeitbezug zu beachten, so sind zeitgewichtete als auch ungewichtete Ergebnisdaten möglich. Beispielsweise spielt üblicherweise bei Maximal- und Minimalwerten die Zeit eine untergeordnete Rolle, zur Ermittlung der Mittelwerte können aber zeitgewichtete Daten benötigt werden.

[7] n ist die Anzahl der Simulationsläufe.
X stellt ein Merkmal dar.
x_i ist ein Beobachtungswert für das Merkmal X.
p_i ist die Wahrscheinlichkeit, dass das Merkmal X den Wert x_i annimmt.
[8] Mittelwert für ordinal skalierte Daten [BV2008, S. 29]
[9] Quartil Wert unter bzw. über dem 25% der beobachteten Werte liegen [BV2008, S. 35f]
[10] Vertrauensintervalle „[…] mit welcher Wahrscheinlichkeit der wahre statistische Parameter […] in diesem Intervall liegt“ [We2008, S. 144].

Wie bereits beschrieben, wird in den wenigsten Fällen direkt auf Rohdaten zu Analyse- oder Vergleichszwecken zurückgegriffen, sondern meist mit verdichteten Informationen in Form von Kennzahlen gearbeitet. Kennzahlen sind ein in vielen Bereichen verbreitetes und bei Weitem nicht auf die Auswertung von Simulationsmodellen beschränktes Instrument. So existiert gerade im Kontext Produktions- und Logistiksysteme ein breites Anwendungsspektrum von logistischen Kennzahlen [Be2008, S. 181-184]. Auf Möglichkeiten des Einsatzes, Funktionen usw. von logistischen Kennzahlen (vgl. z.B. [Be2008], [SRS2010], [Ho2012]) außerhalb der Simulation soll an dieser Stelle nicht näher eingegangen werden, wohl aber muss auf die Kennzahlenkataloge hingewiesen werden. Diese Kataloge sind ein nicht zu unterschätzendes Hilfsmittel, um die Auswahl geeigneter, auch außerhalb der Simulation erhebbarer, Kennzahlen zu unterstützen. Die Kennzahlenkataloge geben Hinweise auf die Eignung einzelner Kennzahlen und machen Vorgaben bzgl. der Berechnung und zur Interpretation der Grunddaten sowie der Benamung. Dies führt ggf. zu einer verbesserten Kommunizierbarkeit der Ergebnisse, verbesserten Vergleichbarkeit zwischen Simulationsstudien und/oder des modellierten Realsystems usw. [Ho2012].

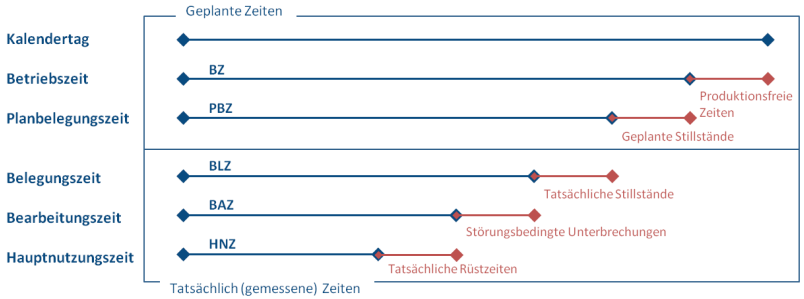


Abbildung 6: Zeitmodell für Produktionseinheiten nach [VDMA 66312-1, S.10]

Beispielhaft sei das VDMA Einheitsblatt 66412-1 "Manufacturing Execution Systems (MES) - Kennzahlen" [VDMA66412-1] genannt, das einen geeigneten Kennzahlenkatalog darstellt, der auch im Rahmen der Simulation von Produktions-und Logistiksystem herangezogen werden kann. Die Definitionen dieses Kennzahlenkatalogs sind in dem im Abschnitt 3.4.5 beschriebenen Prototypen zur Experimentauswertung eingeflossen. Grundlage für die Definition konkreter Kennzahlen ist ein einheitliches Verständnis von Basiswerten, so werden zunächst Zeit- und Mengenmodelle definiert, ein Auszug des Zeitmodells für Produktionseinheiten gibt Abbildung 6 wieder.

Aufbauend auf diesen eindeutigen Zeit- und Mengenmodellen werden entsprechend Kennzahlen definiert, z.B. der Belegungsgrad als Quotient aus Belegungszeit und Planbelegungszeit (vgl. [VDMA 66412-1, S. 17]).

Des Weiteren soll hier ergänzend auf drei weitere Kataloge hingewiesen werden: auf die VDI Richtline 4400 "Logistikkennzahlen für die Produktion" [VDI4400-2], die ISO Norm "Automation systems and integration - Key performance indicators (KPIs) for manufacturing operations management" [ISO22400-2] sowie das Wiki "Begriffe & Kennzahlen in Produktion und Logistik" [We2012].

2.2 Automatische datengetriebene Modellgenerierung und Initialisierung

Wie bereits in vorhergehenden Abschnitten dargelegt, sind die größten Potentiale für eine weiter wachsende Verbreitung der Simulation von Produktionssystemen, auch in Klein- und Mittelständischen Unternehmen (KMU), darin zu sehen, dass: 1. der Aufwand und die benötigte Zeit für erfolgreiche Simulationsstudien verringert wird, 2. gerade nicht ausgesprochene Simulationsexperten befähigt werden, Simulationsmodelle zu erstellen oder zumindest zu nutzen und 3. die Simulation sowohl technisch als auch organisatorisch besser in die betrieblichen Abläufe und die betriebliche IT-Infrastruktur integriert wird.

Verschiedene Erhebungen haben gezeigt, dass grundsätzlich erheblicher Aufwand für die Datenbeschaffung, -aufbereitung und Modellerstellung anfällt. Nach Acel [Ac1996, S.15] entfallen allein auf die Datenerhebung und die Modellerstellung über 50% des zeitlichen Aufwands, neuere Untersuchungen von Müller-Sommer [MS2010, S.1ff; MS2013, S.4-5] im Kontext der Belieferungssimulation im Automotiv Sektor zeigen ähnliche Werte (siehe Abbildung 7). Somit ist gerade auch in diesen Phasen mit erheblichem Einsparpotential zu rechnen.

Die (teil-) automatische datengetriebene Modellgenerierung und -initialisierung verspricht die genannten Ziele und Potentiale durchweg zu unterstützen.

Allgemein wird unter dem Begriff der (teil-)automatischen Modellgenerierung im Simulationskontext ein Ansatz verstanden, bei dem die Simulationsmodellerzeugung nicht manuell mit den Modellierungswerkzeugen bzw. der Programmierumgebung des Simulators erfolgt, sondern vielmehr aus externen Datenquellen, mittels Schnittstellen und Algorithmen, generiert wird. Im engeren Sinn spricht man daher auch von datengetriebener Modellgenerierung [SBM2010, S35f; BS2010b, S.545; We2009, S. 7; Ec2002, S.7ff].

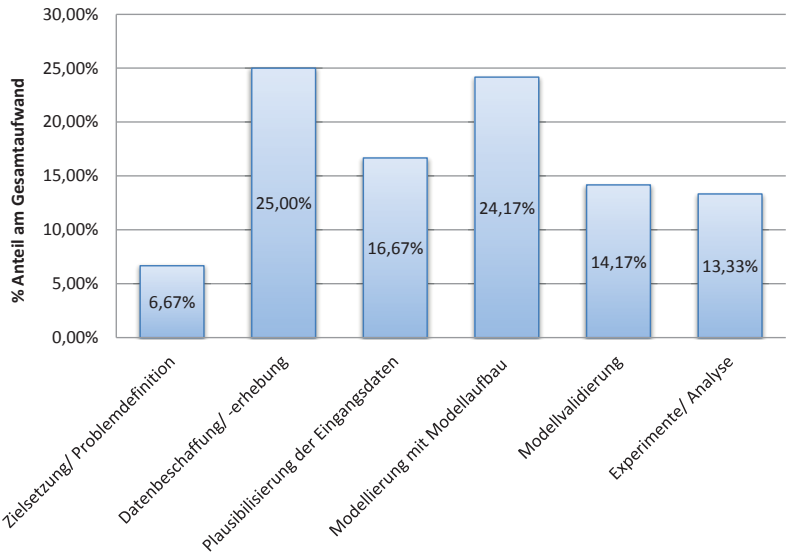


Abbildung 7: Aufwände der Belieferungssimulation (in Anlehnung an [MS2010])

Von denen im Folgenden näher betrachteten Ansätzen der datengetriebenen Modellgenerierung sind Methoden abzugrenzen, die zwar ebenso zur Modellerstellung auf die Modellierungswerkzeuge bzw. Programmierumgebung des Simulators verzichten aber nicht auf bestehende Datenbestände zurückgreifen, sondern auf der Nutzung von Modellbeschreibungssprachen basieren, also eine Modelltransformation stattfindet. Im Zuge dieser Ansätze kommen eigens erstellte oder im Unternehmen bereits bestehende Modelle zum Einsatz um Systeme simulatorneutral zu beschreiben und anschließend Simulationsmodelle zu erzeugen. So nutzen beispielsweise Kloos und Nissen [KN2010; Kl2013] einen transformationsregelbasierten Ansatz um bestehende Geschäftsprozessmodelle, z.B. Business Process Model and Notation (BPMN) Modelle oder Erweiterte Ereignisgesteuerte Prozesskette (eEPK), in Simulationsmodelle zu überführen. Andere Ansätze nutzen beispielsweise die Specification and Description Language (SDL) [FC2010] oder SysML [SR2010; SR2011] als Beschreibungssprache.

Auch im Rahmen des in dieser Arbeit vorgestellten CMSD basierten Ansatzes werden Transformationen durchgeführt, zunächst werden Rohdaten in einem (formalen) Konzeptmodell abgebildet welches anschließend in das konkrete Simulationsmodell überführt wird. Hierbei herrscht aber immer eine rein datengetriebene Sichtweise vor, wobei auf die Extraktion aus vorhandenen Datenquellen fokussiert wird.

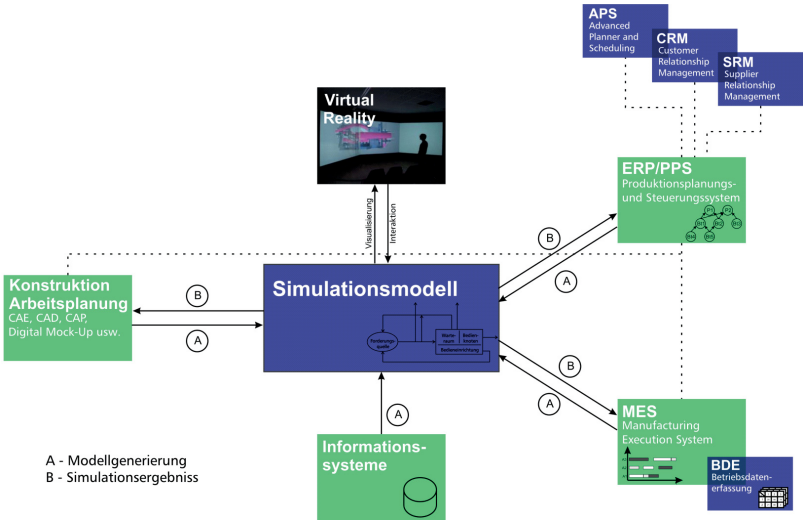


Abbildung 8: Relevante Datenquellen für die automatische Modellgenerierung [BS2010a, S.2]

Für die datengetriebene automatische Modellgenerierung existiert ein breites Spektrum an potentiell geeigneten externen Datenquellen und zugehörigen IT-Systemen, eine Auswahl ist in Abbildung 8 abgebildet, wobei je nach Anwendungsfall der Simulation und in Abhängigkeit des Unternehmens nicht alle aufgeführten Systeme vorhanden oder geeignet sind bzw. auch weitere IT-Systeme betrachtet werden müssen, womit sich eine erhebliche Zahl potentieller, zum Teil bidirektionaler Schnittstellen ergibt.

Nach Eckardt [Ec2002] lassen sich die Ansätze der automatischen datengetriebenen Modellgenerierung grob in 4 Klassen einteilen (siehe Abbildung 9), dabei sind im Kontext der Simulation mathematisch statistische Ansätze von vornherein auszuschließen.

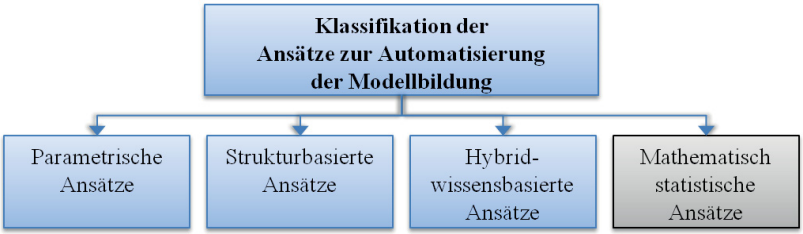


Abbildung 9: Klassifikation von Ansätzen zur automatischen Modellbildung nach [Ec2002, S.39]

Die drei relevanten Klassen für die Automatisierung der Erstellung diskret-ereignisgesteuerter Simulationsmodelle lassen sich wie folgt zusammenfassend beschreiben:

1. **Parametrische Ansätze**: Kennzeichnend für diesen Ansatz ist, dass Modelle auf Basis existierender, in Bibliotheken gespeicherter Simulationsbausteine erstellt werden, die vom Modellgenerator auf Basis von Parametern selektiert und konfiguriert werden.
2. **Strukturbasierte Ansätze**: Ausgangspunkt der Modellgenerierung bilden in diesem Ansatz Daten, die die Struktur des abzubildenden Systems beschreiben. Hierzu zählen lt. Eckardt insbesondere Fabriklayoutdaten aus entsprechenden CAD-Systemen.
3. **Hybrid-wissensbasierte Ansätze**: Diese Ansätze kombinieren Verfahren der künstlichen Intelligenz (Expertensysteme, neuronale Netze u.ä.) mit den beiden bisher dargestellten Verfahren.

Kritisch zu hinterfragen ist aber, ob eine klare Zuordnung der aktuellen Arbeiten zum Thema Modellgenerierung zu einer Klasse möglich und vor allem auch zielführend ist. So kommen oft sowohl strukturelle Daten, insbesondere Layoutinformationen, parallel zu Parameterdaten zum Einsatz. Besonders deutlich ist dies im Kontext der Digitalen Fabrik zu beobachten, wo geometrische Informationen nicht nur zum Zweck der Simulation durch zusätzliche Parameter ergänzt werden. Somit könnte die überwiegende Zahl der aktuellen automatischen Modellgenerierungsverfahren[11] als „hybrid“ klassifiziert werden, bei einem nicht unerheblichen Teil besteht zusätzlich der Anspruch Expertenwissen nutzbar zu machen, womit man oft auch von hybrid-wissensbasierten Ansätzen sprechen kann [SBM2010, S. 36f].

[11] z.B. [MS2012, Se2005, Ze2006, Je2007]

Neben der Klassifikation nach Eckardt und der in vielen Veröffentlichungen genutzten Unterscheidung nach Unterstützung der Planungs- oder Betriebsphase können Modellgenerierungsansätze nach einer Vielzahl weiterer Kriterien analysiert und klassifiziert werden, eine Auswahl möglicher Kriterien und Ausprägungen ist in [SBM2010, S. 37] in Form eines morphologischen Kastens aufgelistet (vgl. Tabelle 4).

Tabelle 4: Klassifikationsmöglichkeiten von Modellgenerierungsansätzen [SBM2010]

Kriterium	*Ausprägung*				
Einsatzfall der generierten Simulations-modelle	planungsbegleitend		betriebsbegleitend	hybrid	
	strategisch (z.B. Standort)	taktisch (z.B. Layout, Puffergrößen)			
Fertigungstyp	Job-Shop	Flow-Shop	Open-Shop		
Fokussiertes Gewerk	Montage	Lackiererei	Logistik	Förder-technik	Andere
Nutzergruppe	Fachabteilung	Simulationsexperte	Jeder		
Grad der Automatisierung der Modellerstellung	keine	teilautomatisch		voll automatisch	
		Struktur	Verhalten	beides	
Ansatz der Modellerstellung	Standardisierte Eingabe-formulare	Dialoggeführt (Wizzard o.ä.)	Referenz-modelle	Direkter generischer Modellaufbau	
				eine Daten-quelle	alle relevanten Systeme
Unterstützte Phasen der Simulationsstudie	Modellerstellung	Verifikation und Validierung	Experiment / Initialisierung	mehrere	
Technische Umsetzung der Schnittstelle	textbasiert (z.B. *.csv)	tabellen-basiert (z.B. Excel)	XML (z.B. CMSD)	Datenbank (z.B. SQL)	
Art der fachlichen Schnittstelle	layout-basiert (z.B. SDX)	produkt- basiert (z.B. STEP)	prozess-basiert	sonstige / hybrid	
Wieder-verwendung des Modells	keine		mehrmalig		
	keine weiteren Fragestellungen	nicht wirtschaftlich	manuelle Nachbearbeitung	Neuparametrisierung	kontinuierliche Anpassung

So vielfältig die Klassifikationskriterien, so vielfältig sind auch die projektabhängigen Anforderungen an eine automatische Modellgenerierung und -initialisierung. Je nach Anwendungsfall und den aktuell vorherrschenden Rahmenbedingungen des Simulationsprojektes können verschiedenste Anforderungen unterschiedlich priorisiert sein. Nach Eckardt sind 6 primäre Anforderungen für die automatische Modellgenerierung zu identifizieren, die Forderung nach flexiblem Abbildungsbereich, Modellierungsflexibilität, Verfahrensflexibilität, Anwendungsflexibilität, Performance und hohem Automatisierungsgrad [Ec2002, S.36-38]. In allen im Rahmen der Literaturrecherche betrachteten Ansätzen sind analog explizit oder zumindest implizit ähnliche Anforderungen formuliert. Auf Basis der in der Literatur getätigten Aussagen wurden im Rahmen dieser Arbeit folgende allgemeine Anforderungen an die Simulationsmodellgenerierung im Kontext der Produktion formuliert:

- Flexibilität bezüglich der abzubildenden (Produktions-) Systeme;
  Ziel ist es, eine möglichst breite Palette verschiedener Produktionssysteme (technisch, organisatorisch, strukturell und dispositiv) zu unterstützen.
- Flexibilität bezüglich des Detailierungsgrades bzw. des Abstraktionslevels;
  Ziel ist es, das Abstraktionslevel durch Aufgabenstellung und Datenmaterial nicht durch den Modellgenerator bestimmen zu lassen, zudem soll es durchaus möglich sein in einzelnen Subsystemen verschiedene Detaillierungsgrade zu nutzen.
- Flexibilität bezüglich der Simulationsanwendung;
  Ziel ist es, die Modellgenerierung unabhängig vom Simulationszweck zu unterstützen.
- Erweiterungsfähigkeit des Ansatzes;
  Ziel ist, dass Prozess und Unternehmensspezifika jederzeit implementierbar und geeignet in den automatischen Modellaufbau integrierbar sind.
- Automatisierungsgrad;
  Ziel ist es, einen möglichst hohen Automatisierungsgrad zu erreichen, wobei persistente "manuelle" Ergänzungen möglich bleiben sollen.
- Flexibilität bezüglich der Datenquellen;
  Ziel ist es, prinzipiell von der technischen Datenquelle zu abstrahieren.
- Unterstützung der GoM;
  Ziel ist es, die GoM so weit möglich zu unterstützen.

Diese Anforderungen werden im Folgenden zum einen genutzt um die bestehenden Ansätze kritisch zu analysieren, zum anderen muss sich das im Rahmen dieser Arbeit entwickelte Framework ebenso an ihnen messen lassen.

2.2.1 Relevante Ansätze der Modellgenerierung

Bestrebungen zur Automatisierung der Simulationsmodellerzeugung sind als Idee nicht neu und werden seit über 15 Jahren verfolgt. In der Literatur sind eine Vielzahl von Ansätzen zur automatischen Modellgenerierung beschrieben, eine Auswahl der Ansätze, die nach Meinung des Autors wichtige Meilensteine darstellen, werden in den folgenden Abschnitten chronologisch geordnet vorgestellt. "Vielfach konzentrieren sich diese Ansätze auf spezielle Anwendungsbereiche, z.B. auf die Fabrikplanung [...] oder die layoutorientierte Gestaltung von Materialflusssystemen" [Se2005, S.23].

Einer der ersten öffentlich publizierten Ansätze zur Simulationsmodellgenerierung wurde unter dem Titel "Layout based model generation" (LBMG) 1995 von Lorenz und Schulze vorgestellt [LS1995]. Fokus des Ansatzes war die Interpretation von CAD Daten zur Erzeugung von Simulationslayouts, zusätzlich wurde die eigentliche Modellerstellung, z.B. durch die Ermittlung aller relevanten Entitäten unterstützt. Das Grundprinzip des Ansatzes wurde als simulatorneutral bezeichnet, wobei aber simulator- und vor allem domänspezifische Komponenten benötigt werden. Eine Einschränkung auf Produktionssysteme wird nicht vorgenommen, so werden neben JobShop Produktionssystemen auch Dienstleistungsszenarien (Joe's Barbershop Beispiel) und die Verkehrssimulation als Einsatzfeld aufgeführt. Eine weitere Restriktion stellt das Mappen der CAD und Simulationselemente, inklusive zusätzlicher Informationen aus Drittsystemen, anhand des Namens bzw. Typs dar, somit müssen zwingend gemeinsame Namenskonventionen eingehalten werden. Auf die nicht aus dem Layout ableitbaren Informationen (z.B. Arbeitspläne, Ankunftsraten) wird nicht näher eingegangen bzw. eine direkte Abfrage des Nutzers oder Einbindung spezieller Datenbestände vorgeschlagen.

Ebenfalls 1995 publizierte Splanemann seinen Ansatz "Teilautomatische Generierung von Simulationsmodellen aus systemneutral definierten Unternehmensdaten" [Sp1995]. Novum dieses Modellgenerierungsansatzes ist, bereits vorhandene Daten aus betrieblichen Informationssystemen zur Modellerstellung heranzuziehen sowie ein standardisiertes Datenformat zur systemneutralen Modellbeschreibung zu nutzen. Hinzu kommt eine grundlegende und ausführliche Ausführung zu den für die Modellgenerierung von Produktionssystemen benötigten Eingangsdaten. Als Primärdatenquelle dieses Ansatzes ist ebenso das Layout in Form von CAD Daten angedacht, diese werden durch Systemlastdaten aus Produktionsplanungs- und Steuerungssystemen (PPS) oder Werkstattsteuerungssystemen sowie durch Informationen, wie beispielsweise Maschinendaten (z.B. Störprofile aus Datenbanken) ergänzt. Die in den IT-Systemen vorliegenden Daten werden mittels eines Preprozessors in das neutrale STEP basierte Datenformat überführt, aus welchem mittels spezieller simulatorspezifischer Postprozessoren die eigentlichen Simulationsmodelle erzeugt

werden. Neben der grundlegenden Problematik des Mappens von Inhalten der verschiedenen Datenquellen mit ggf. unterschiedlicher Nomenklatur ist besonders die starke Fokussierung auf das Layout kritisch zu hinterfragen. So werden aus der Anordnung der Elemente im Layout Rückschlüsse auf den Fertigungsprozess getroffen. Dies kann allein in den auch von Splanemann primär betrachteten stark verketteten Produktionssystemen, wie der Fließfertigung, einen interessanten Ansatz darstellen.

Ein weiterer relevanter Ansatz stellt die von Wuttke 2000 veröffentliche Dissertation mit dem Titel "Mehrfachnutzung von Simulationsmodellen in der Produktionslogistik " dar [Wu2000], in der die automatische Modellgenerierung ein Teilaspekt des Gesamtkonzeptes darstellt. Erklärtes Ziel von Wuttke ist es, die Simulation in möglichst viele alltägliche Planungsabläufe zu integrieren und zu automatisieren. Dazu wird vor allem die automatisierte Durchführung von Simulationsstudien untersucht. Dazu sind zwei grundlegende Szenarien zu differenzieren: zum einen die Unterstützung des operativen Betriebs, zum anderen die Unterstützung im Rahmen der Fertigungsplanung. Im ersten Fall wird ein bestehendes Simulationsmodell verwendet (parametrisierbare Referenzmodelle), das mittels Eingabemasken automatisiert variiert, d.h. initialisiert und anschließend entsprechend zur Ausführung gebracht wird. Im zweiten Fall der Fertigungsplanung, wenn also meist die Struktur des Modells variiert, wird ein proprietäres Informationsmodell zur Neugenerierung des Simulationsmodells genutzt. Dieses Informationsmodell wird allein über ein Hilfsprogramm gepflegt und nutzt keinerlei Datenbestände der betrieblichen IT-Systeme, allein ein Vorgehensmodell und einfache Eingabevalidierungen sollen die konsistente Eingabe der Daten sicherstellen.

2005 wurde Selkes Dissertation "Entwicklung von Methoden zur automatischen Simulationsmodellgenerierung" publiziert, der darin beschriebene Ansatz fokussiert rein auf dem betriebsbegleitenden, d.h. kurzfristig operativen Anwendungsfall der Simulation im turbulenten betrieblichen Umfeld. Erkannte Hauptanforderungen aus dem turbulenten Umfeld an die betriebsbegleitende Simulation war zum einen die, dass "das zugrunde liegende Simulationsmodell [...] schnell und effizient an die aktuelle Betriebssituation der Produktion anpassbar sein" muss [Se2005, S.22] und zum anderen, dass "die betriebsbegleitende Simulation [...] in die alltäglichen Abläufe der Produktionsplanung und -steuerung integrierbar sein" [Se2005, S.22] muss. Aufbauend auf dem Ansatz von Splanemann kristallisierte sich als eine Forschungsthematik die Identifikation und Beschreibung von Strategien und Abläufen der Produktionssteuerung, die zur automatischen Modellgenerierung nutzbar sind, heraus (vgl. Abschnitt 3.2). Als Datenquellen wurden konsequent Daten aus PPS-Systemen und vor allem Daten der Betriebsdatenerfasssung (BDE-Daten) genutzt. Diese werden gefiltert, gesammelt, exportiert, ggf. händisch um simulationsrelevante Daten ergänzt und in einer Simulationsdatenbank gespeichert. Vorab definierte Strategien werden

automatisch interpretiert, indem nach einzelnen Mustermerkmalen in den BDE-Daten Regeln erkannt und entsprechend abgelegt werden. Trotz der Anforderung an die schnelle Anpassbarkeit wird eine Adaption von bereits generierten Modellen nicht thematisiert, das Vorgehen sieht hier nur eine möglichst schnelle Neugenerierung und Initialisierung vor. Selke bemerkt in seiner Publikation, dass sein Prototyp keinesfalls für den produktiven Einsatz geeignet ist, aber dass die automatische Modellgenerierung konsequent integriert in die betriebliche IT den Aufwand zur Modellerstellung deutlich senken und in den meisten Fällen auf Simulationsexperten verzichtet werden kann.

Ein weiterer Ansatz, der ebenso wie die beiden vorherigen Ansätze nicht allein auf die Modellgenerierung abzielt, wurde 2006 von Zenner in seiner Dissertation "Durchgängiges Variantenmanagement in der Technischen Produktionsplanung" veröffentlicht [Ze2006]. So wird durch Zenner die Abbildung von Varianten in komplexen Produktionsprozessen in Form von Prozessgraphen thematisiert, wobei abschließend eine Überführung in die Simulation unterstützt wird. Im Zuge der eigentlichen Simulationsmodellgenerierung werden bestehende Ansätze um an die Variantenfertigung angepasste generische Varianten- und Logistikbausteine ergänzt. Novum bzgl. der Modellgenerierung stellen aber die Nutzung von XML als ein mögliches neutrales Datenaustauschformat, neben einer tabellenbasierten Darstellung, sowie die Konzeption und (Teil-) Implementierung einer bidirektionalen Schnittstelle dar. Im Gegensatz zu den meisten anderen Ansätzen wird auf das Einbeziehen von Layoutinformationen vollständig verzichtet, die Anordnung der Elemente in der Simulation basiert allein auf einer Heuristik zur überschneidungsfreien Anordnung, womit eine Validierung des Modells erschwert wird.

Ein weiteren Meilenstein der Simulationsmodellgenerierung ist in dem 2007 veröffentlichten Ansatz von Jensen "Eine Methodik zur teilautomatischen Generierung von Simulationsmodellen aus Produktionsdatensystemen am Beispiel einer Job Shop Fertigung" [Je2007] zu sehen, in diesem wird konsequent auf XML zum Datenaustausch gesetzt. Fokus liegt in dieser Arbeit auf der Zusammenführung von Daten aus unterschiedlichen betrieblichen Informationssystemen, dazu wurde ein Datenframework zur rudimentären Abbildung einer Auswahl von Simulationsentitäten in XML definiert und entsprechende Anbindungsmodule für einzelne Datenquellen entworfen. Aufbauend auf den Architekturansatz der Service Orientierte Architekturen (SOA) wurde eine webbasierte Middleware Lösung entworfen und implementiert, die flexible Datenanfragen managen kann, indem sie die Services entsprechend orchestriert. Ebenso wie Wuttke sieht Jensen eine Nachbearbeitung der Daten in entsprechenden Masken einer grafischen Oberfläche vor. Eine prototypische Umsetzung der Modellgenerierung erfolgte mittels "Parser" für eine Auswahl von Simulatoren.

Die genannten Ansätze sind nur eine Auswahl der nach Meinung des Autors wichtigsten Veröffentlichungen im Kontext der automatischen Modellgenerierung, eine Vielzahl weiterer Arbeiten, die sich zumindest mittelbar mit der automatischen Generierung von Simulationsmodellen auseinandersetzen, wurden in den letzten 15 Jahren veröffentlicht. So z.B. beschreibt Rooks 2009 [Ro2009] in seiner Dissertation die automatische Generierung von Simulationsmodellen zum Zweck der Logistiksimulation. Eine Erkenntnis war es, dass die bestehende Datenbasis nicht ausreichend ist, um ein valides Modell zu erstellen. Anders als die meisten anderen Autoren, z.B. Wuttke oder Jensen, wurde von Rooks nicht allein ein zusätzliches Hilfsprogramm, sondern eine gezielte Anreicherung der Daten in den Planungssystemen präferiert. Diesen Ansatz kritisch hinterfragend, entwarf Müller-Sommer in seiner Arbeit [MS2013] ein Simulationsdatengerüst, das regelbasiert zusätzliche Daten planungsfallneutral zur Verfügung stellt, sowie weitere Maßnahmen zur Datenqualitätssicherung in Form von Filterklassen. Weitere Ansätze sind bspw. veröffentlicht in [Fr2003], [Wa2003] oder [Wu2008]. Im englischsprachigen Raum sind Ansätze zur datengetriebenen Modellgenerierung weniger verbreitet, hier wird Modellgenerierung vor allem auf Basis der Kombination bestehender Komponenten betrieben ("automated reuse of model components") [LZ1996; So2000; SR2001; Gy2008; HSV2011].

Generell ist aber zu konstatieren, dass "eine allgemeingültige, allumfassende Lösung zur Automatisierung der Modellbildung für den operativen Produktionsprozess [...] bisher weder gefunden noch konzeptionell vollständig entwickelt" wurde [Ec2002, S.50]. Kein Ansatz erfüllt alle Anforderungen an vollständig automatisierte und produktionssystemunabhängige Modellgenerierung, vor allem die Generierung von Steuerungsstrategien wird zumeist vernachlässigt [Se2005, S.30; BRA2008, S. 441]. Die Problematik der Steuerungsstrategien wird in einem separaten Abschnitt 3.2 aufgegriffen.

In seiner Dissertation schreibt Selke: „In sehr vielen Ansätzen konnte gezeigt werden, dass die automatische Modellgenerierung im Rahmen der betriebsbegleitenden Simulation geeignet ist, die Effizienz bei der Erstellung von Simulationsmodellen zu erhöhen. Vielfach fehlen den Ansätzen Allgemeingültigkeit, dennoch kann festgehalten werden, dass die Kopplung mit anderen betrieblichen Informationssystemen ein wesentlicher Bestandteil der automatischen Modellgenerierung sein muss.“ [Se2005, S.29f]. Diese Aussage bzgl. prinzipieller Eignung und die Forderung der Integration in die IT-Infrastruktur kann zweifelsfrei auch für nicht betriebsbegleitende Ansätze propagiert werden. Die Kopplung zu betrieblichen Informationssystemen kann nach Meinung des Autors auf Grund der Vielzahl möglicher Schnittstellen (vgl. Abbildung 8) allein mittels eines neutralen standardisierten Formats realisiert werden. So stellte auch Spieckermann fest, dass systemneutrale Austauschformate einen wesentlichen Beitrag

zur weiteren Verbreitung der rechnergestützten Modellgenerierung leisten können [Sp2005, S.6]. Hierbei muss nicht nur die Erstellung des Modells sondern ebenso die Adaption, Nutzung des Modells sowie die Ergebnisanalyse abgedeckt werden. Hierzu sind Fragestellungen bzgl. geeigneter Experimentplanung, -verteilung und -auswertung zu betrachten, aber vor allem ist eine adäquate Initialisierungsmöglichkeit (vgl. Abschnitt 2.2.2) vorzusehen. Eine weitere zu diskutierende Kernfragestellung der automatischen Modellgenerierung ist die Repräsentation von dynamischem Verhalten bzw. Steuerstrategien (vgl. Abschnitt 3.2) sowie die Auswertung und Repräsentation von Ergebnisdaten einzelner Simulationsläufe und ganzer Simulationsexperimente (vgl. Abschnitt 2.1.7).

2.2.2 Initialisierung von Simulationsmodellen

Wie bereits angedeutet ist die Automatisierung der Simulation nicht allein auf die Modellgenerierung zu beschränken, als weitere wichtige Komponente ist für einige Anwendungsszenarien ebenso eine automatisierte Simulationsmodellinitialisierung zu sehen.

Unter Initialisierung versteht man im Kontext der DES dabei die Art und Weise, wie vor dem eigentlichen Ausführen des Simulationsmodells die modellinternen Kontrollstrukturen mit Startwerten belegt werden, so dass nach erfolgter Initialisierung des Simulationsmodells der gegenwärtige Zustand eines realen (oder gedachten) Systems mit hinreichender Genauigkeit, z.B. zu Prognosezwecken repräsentiert wird [Ho2007, S. 78f].

In den diskutierten Modellgenerierungsansätzen wird die Initialisierung nicht explizit thematisiert. In allen rein planungsbegleitenden Ansätzen resultiert dies aus der gängigen Vorgehensweise, dass mit "leeren", betriebsbereiten und störungsfreien Modellzuständen gestartet wird, wobei die Auswertungen erst auf Datenerhebungen nach einer angemessenen Einschwingphase basieren [HST2005, S. 1794]. Für betriebsbegleitende Ansätze ist das Arbeiten mit Einschwingphasen aber ungeeignet.

In der Online bzw. Symbiotic Simulation (vgl. [Hu2006; Ay2008a; Ay2008b]) ist die Initialisierung ein immanenter Teil des Konzeptes, verschiedene Initialisierungsverfahren aber auch Voraussetzungen, z.B. bzgl. Datenqualität, werden diskutiert. So werden von Hanisch, Schulze und Tolujew in [HST2005] Initialisierungsansätze für die Onlinesimulation analysiert, wobei ein Ansatz die automatische Modellgenerierung mit Initialisierung darstellt, dieser Ansatz wird auch im Rahmen dieser Arbeit präferiert. Weitere Ansätze basieren auf der Synchronisation eines Hauptmodells (parent model) und der Abspaltung von entsprechenden Kindmodellen bei Bedarf, wobei verschiedene Spielarten sowohl bei der

Synchronisation, der permanenten oder zyklischen Anpassung des Zustand als auch bei der Abspaltung diskutiert werden.

Grundlage für jegliche Initialisierung bildet ein u.U. automatisch generiertes, ausreichend genaues Simulationsmodell, welches geeignete Möglichkeiten zur simulationsinternen Parameterbelegung bietet, dies soll im folgenden als gegeben angenommen werden. Eine weitere Grundbedingung stellen die hinreichend guten Last- bzw. Zustandsdaten des Realsystems, z.B. aus der Maschinen- bzw. Betriebsdatenerfassung dar [BSS2011a, BSS2011b]. Tabelle 5 gibt Beispiele möglicher, für die Initialisierung zusätzlich benötigter Informationen wieder.

Tabelle 5: Kategorien von Initialisierungsdaten (in Anlehnung an [BSS2011a, S.2232])

Daten über		**Beispiele für Informationen**
Ressourcen	Maschine/ Bearbeitungs-stationen	Status (Idle, working, setup, paused, failed, …), Rüstzustand ...
	Werker	Status (working, paused, …), Ort...
	Fördereinrichtung	Status (Idle, working, paused, failed ...), aktuelle Geschwindigkeit, Ort, Typ …
Aufträge		Aktueller Prozessschritt, Status, Ausschuss, Typ, …
Teile		Ort, Status
Systemzeit / Experimentzeit		

Wie zu sehen werden für die Initialisierung, je nach gewünschter Detaillierung, vollständige Informationen über den aktuellen Zustand aller Objekte des Systems benötigt. Dazu zählen vor allem der Zustand, Bearbeitungsfortschritt und der physische Ort aller Aufträge sowie die aktuellen Zustände aller Werker und Maschinen.

Die Detaillierung der für die Initialisierung benötigten Informationen kann je nach Simulationszielstellung sehr hoch sein. So werden abhängig vom gewählten Abstraktionslevel viele Detailinformationen, z.B. der aktuelle Aufenthaltsort eines Werkers, der konkrete Ort und Zustand eines Förderers (z.B. Gabelstapler, Milktrain) oder die sekundengenaue Restbearbeitungszeit eines schon begonnen Arbeitsganges relevant und müssen betrachtet werden, da ein Verzicht bzw. eine Fehlbelegung zu relevanten Abweichungen vom Systemverhalten führen kann.

Informationen bezüglich einzelner Ressourcen bilden die erste Gruppe von Initialisierungsdaten. Für die wohl in allen Produktionssystemen vorkommende Klasse der Bearbeitungsstationen (z.B. Maschinen aber auch Handarbeitsplätze), ist der (Betriebs-) Status sowie der aktuelle Rüstzustand essentiell. Diese beiden Attribute sind

in der Regel im Simulationsmodell leicht abbild- und modifizierbar. Es lassen sich sechs Hauptstatus für Bearbeitungsstationen identifizieren:

- "frei" (idle),
- "beschäftigt" (busy),
- "rüstend" (setup),
- "gestört" (broken/failed),
- "pausierend" (paused) und
- "in Wartung" (under maintenance).

Weitere Detaillierungen sind denkbar, so kann z.B. zusätzlich im Status "gestört" der Störgrund oder die geschätzte Stördauer relevant sein. Dagegen sind Informationen welche Aufträge, Werkstücke usw. im Verfügungsbereich der Station vorkommen meist an den entsprechenden Objekten, nicht der Station zu modellieren [BSS2011a, S.2232].

Die Bearbeitungsstationen/Maschinen sind typischerweise immobil, für weitere ggf. mobile Ressourcenklassen müssen weitere Informationen bereitgestellt werden. Diese Ressourcenklassen können zum einen zusätzliche Informationen benötigen, so kann für Werker oder auch Förderer die aktuelle Position/Stellung innerhalb des Fertigungssystems relevant sein. Zum anderen können auch Attribute anders interpretiert oder spezifische Wertebereiche zugelassen werden. So hat ein Werker wohl einen Status der analog der Bearbeitungsstation Werte wie "frei", "beschäftigt" oder "pausiert" annehmen kann, aber in der Regel nicht die Werte "gestört", "rüstend" oder "in Wartung", gegenüber der stationären Bearbeitungsstation ist aber der Wert "in Bewegung" (in movement) erlaubt [BSS2011a, S.2232]. Zudem sind Informationen zum Werker oft erst in der Verknüpfung zum Aufenthaltsort / Arbeitsplatz vollständig interpretierbar, so kann beispielweise gefolgert werden, dass ein "beschäftigter" Werker der sich an einer "gestörten" Station befindet mit der Reparatur dieser beauftragt ist.

Eine weitere mobile Ressourcenklasse ist die der Förderer, diese ist eine sehr heterogene Klasse in der unterschiedliche Fördersysteme zusammengefasst werden. So reicht die Palette von Staplern über Routenzüge bis zu Fließbändern, somit ist ein großer Spielraum möglicher Initialisierungsattribute und Werte vorhanden. Je nach Abstraktionslevel können Förderer teilweise als Bearbeitungsstationen oder als Werker abstrahiert werden, aber es können ebenso zusätzliche Informationen wie die aktuelle Geschwindigkeit, Beschleunigung, Art, Lage und Anzahl der Ladungsträger benötigt werden. Eine abschließende Betrachtung aller möglichen Förderer ist nicht möglich, wobei zu bemerken ist, dass die Erweiterbarkeit als grundlegende Anforderung an das im Rahmen dieser Arbeit entwickelte Framework auch bei Förderern Anwendung findet [BSS2011a, S. 2232].

Als das Kernobjekt der Initialisierung ist der Auftrag, als veränderliches/bewegliches Objekt des realen Systems, zu sehen. Ohne eine ausreichend genaue Abbildung der Aufträge sind viele, gerade betriebsbegleitende, simulationsbasierte Anwendungen nicht realisierbar. Die Basisinformation, die für die Initialisierung von Aufträgen zwingend benötigt wird, ist das Wissen über den aktuellen Bearbeitungs-/Prozessschritt und dessen Status, d.h. es muss bekannt sein, ob der Auftrag im System aktuell als

- "freigegeben" (released)
- "gestartet"(started)
- "abgebrochen" (cancelled)
- "blockiert" (blocked)oder
- "beendet" (completed)

gekennzeichnet ist und somit ob und wie der Auftrag zu initialisieren ist. Mögliche Übergänge sind Abbildung 10 zu entnehmen. So sind beendete Aufträge für die Initialisierung in der Regel nicht relevant (ggf. Initialisierung der Datenaufzeichnung) und freigegebene, nicht gestartete Aufträge nur in der Auftragsliste zu hinterlegen.

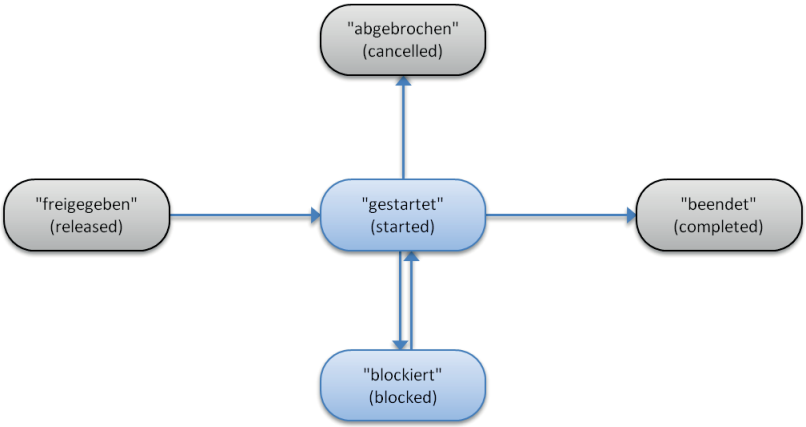


Abbildung 10: Auftragsstatus und mögliche Übergänge

Die tatsächliche Lage eines Auftrags muss nicht explizit initialisiert werden, da in Verbindung mit dem verknüpften Arbeitsplan/Prozessplan und den Initialisierungsdaten der Ressource die Lage bestimmt werden kann, beispielsweise kann in der Regel über den Arbeitsplan die Maschine ermittelt werden, auf der die nächste Bearbeitung des Auftrags erfolgen muss. Befindet sich der Auftrag im Status "gestartet" (started) würde die Ressource vom Auftrag aktuell belegt, befindet sich der Auftrag im Status "blockiert" (blocked) würde sich der Auftrag im Eingangspuffer der Maschine befinden.

Weitere Informationen können wiederum abhängig vom Abstraktionslevel, dem zugrundliegenden System und den Simulationszielen bzw. der Anwendung benötigt werden. So können Attribute wie z.B. Restbearbeitungszeiten (vgl. Abbildung 12) oder aktuelle Ausschussanteile relevant werden. Des Weiteren können auch analog Informationen zu einzelnen Teilen, Baugruppen oder Produkten interessant werden [BSS2011a, S2232].

Eine spezielle Teilkomponente der Initialisierung, die in der Literatur meist nicht explizit diskutiert wird, aber für die praktische Simulationsnutzung in einigen Szenarien als unverzichtbar anzusehen ist, ist die Initialisierung der Datenaufzeichnung/Statistik. Grundlegend sollte sichergestellt werden, dass alle Aufträge, also auch zum Simulationsstartzeitpunkt bereits beendete bzw. vor allem bereits begonnene Aufträge, bei einer Auswertung der Ergebnisse völlig gleichartige Daten liefern wie nach Simulationsstart freigegebene Aufträge. So ist beispielsweise sicherzustellen, dass die tatsächliche Durchlaufzeit (DLZ) eines Auftrags immer korrekt ermittelbar ist (vgl. Abbildung 11). Hierbei können Daten zu bereits beendeten Aufträge (vgl. Abbildung 11; Auftrag C) unter Umständen auch vollständig ggf. auch vorverarbeitet aus Quellsystemen, z.B. PPS oder MES-Systemen, bezogen werden. Der Hauptaugenmerk liegt hier auf bereits begonnen Aufträgen (vgl. Abbildung 11; Auftrag B). Für diese liegen in Drittsystemen auf keinen Fall vollständige Daten vor, ebenso würde eine Simulation ohne entsprechende Initialisierung der Statistik keine bzw. falsche Ergebnisse liefern, in dem z.B. fälschlicherweise der Simulationsstartzeitpunkt als Freigabezeitpunkt des Auftrags angenommen wird, womit in dem hier aufgeführten Beispiel eine deutlich zu kurze DLZ vorgetäuscht würde.

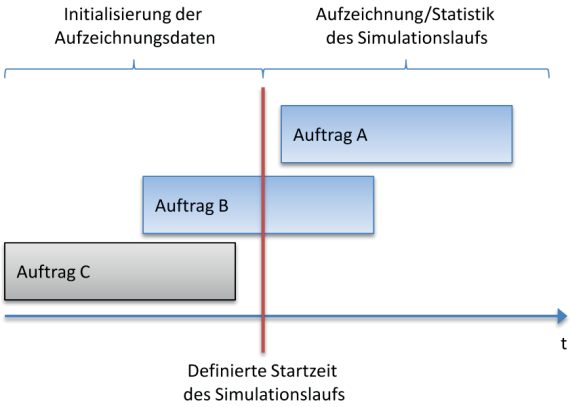


Abbildung 11: Beispiel für Initialisierung von Statistik-/Aufzeichnungsdaten

Durch die konsequente Umsetzung des Konzeptes wird zusätzlich auch die Genauigkeit bzgl. bereits begonnener Operationen ohne den Bedarf für zusätzliche Daten verbessert. So ist, bei Wahrung einer hohen Genauigkeit, keine Angabe von Restbearbeitungs- oder erwarteter Reststörzeiten nötig, wenn der Start der Operation bzw. der Störung bekannt gemacht wird (vgl. Abbildung 12).

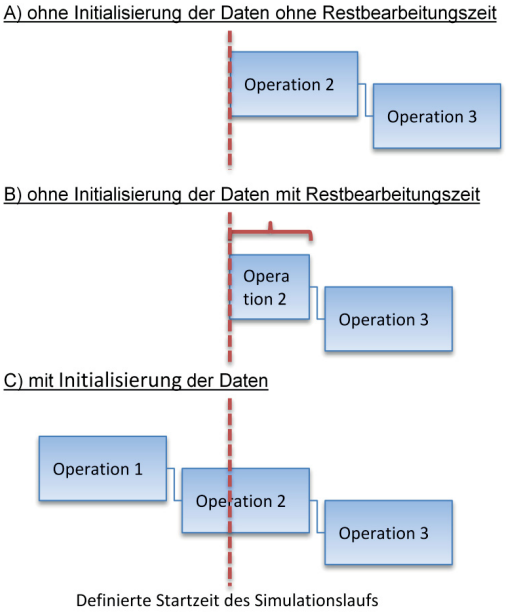


Abbildung 12: Varianten der Initialisierung begonnener Aufträge

Anzumerken ist, dass im Rahmen der Automatisierung der Simulation allgemein aber besonders in der Phase der Initialisierung darauf zu achten ist, dass die Daten dauerhaft in entsprechender Qualität aber auch mit vertretbaren Kosten zu erheben sind, auf Aspekte der operativen Datenerhebung bzw. Betriebsdatenerfassung, wird im Rahmen dieser Arbeit nicht vertiefend eingegangen.

2.2.3 Adaption im Rahmen der Simulation

Neben der Modellgenerierung und Initialisierung ist die Adaption eine weitere zu betrachtende Komponente des hier vorgestellten Ansatzes. Der Begriff Adaption leitet sich vom lateinischen "adaptare", das so viel wie anpassen bedeutet, ab. Im Folgenden wird unter Adaption die Anpassung von Modellen an veränderte Bedingungen ohne vollständigen Neuaufbau verstanden [Be2011, S.14].

Abzugrenzen ist die in diesem Abschnitt thematisierte Adaption im engeren Sinn sowohl von der Initialisierung, die allein den aktuellen Systemzustand betrachtet (definiert über die Zustände einzelner Entitäten und Lastdaten), als auch von der Kalibrierung, welche oft im Zuge der V&V die Feineinstellung einzelner Parameter zum Ziel hat (vgl. [La2007, S.263]). Betrachtet man die Adaption im weitesten Sinn ist hingegen die Kalibrierung als eine Teilmenge der Adaption zu sehen, auch kann die Initialisierung mit einem neuen Zustand im weitesten Sinn als eine Adaption aufgefasst werden.

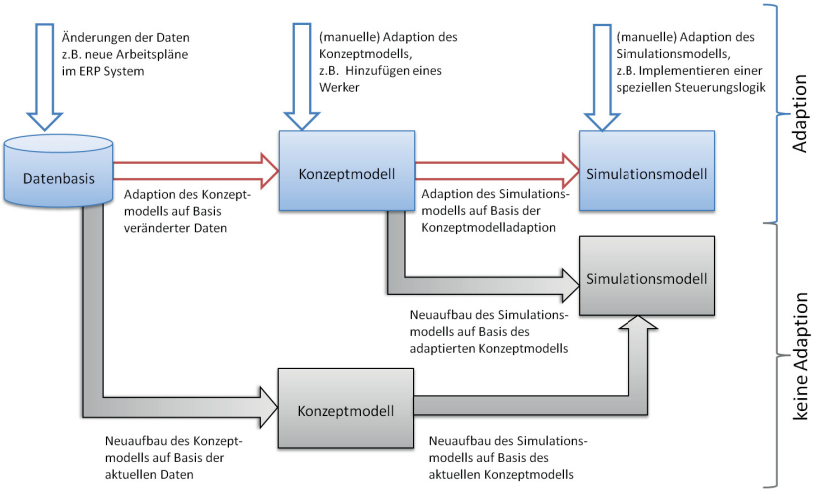


Abbildung 13: Möglichkeiten der Adaption von Konzept- und Simulationsmodellen

Zur Adaption von Simulationsmodellen gibt es im Gegensatz zur Adaption von beispielsweise Geschäftsprozessmodellen (z.B. [BDK2004]) bisher kaum Veröffentlichungen. Prinzipiell kann das Konzept der Adaption im Kontext der Simulation, wie sie in dieser Arbeit betrachtet wird, auf sowohl das Konzeptmodell als auch auf das Simulationsmodell selbst angewendet werden (vgl. Abbildung 13).

Die Adaption des Konzeptmodells kann zum einen auf Änderungen der Datenbasis, z.B. im Zuge der Abbildung weiterer Arbeitspläne in einem ERP-System, beruhen und ggf. (teil-) automatisiert werden oder direkt durch den Simulationsanwender, z.B. mit dem Ziel der Alternativenentwicklung, durchgeführt werden. Die reine Änderung der Eingangsdaten und der Neuaufbau des Konzeptmodells auf diesen angepassten Daten stellt hingegen keine Adaption dar. Die Adaption des Simulationsmodells kann ebenso in zwei unterschiedlichen Spielarten auftreten, zum einen kann eine manuelle Adaption des Simulationsmodells direkt erfolgen, zum anderen kann eine Konzeptmodelladaption zu einer Adaption eines darauf beruhenden Simulationsmodells führen. Abzugrenzen ist

hier wiederum der vollständige Neuaufbau des Simulationsmodells auf Basis eines neuen oder auch adaptierten Simulationsmodells. Im Rahmen der automatischen Modellgenerierung wird Adaption vor allem zur Alternativenbildung sowie für Anpassungen an Änderungen des Realsystems genutzt. Im Fall der Alternativenbildung wird zumeist das Konzeptmodell adaptiert und basierend auf diesem werden neue Simulationsmodelle generiert. Im Fall der Anpassung an Realweltänderungen ist typisch, dass sowohl Konzeptmodell als auch das darauf aufbauende Simulationsmodell adaptiert werden, dies sichert vor allem auch, dass ggf. durchgeführte (manuelle) Adaptionen des Simulationsmodells, in Form von z.B. nicht autogenerierter Steuerungslogik, erhalten bleiben.

Die besondere Herausforderung der Adaption ist hierbei, sämtliche Elemente und deren Verhalten, die nicht gezielt adaptiert werden, unverändert zu belassen [Kl2010, S. 111]. In den hier betrachteten Modellen können theoretisch sämtliche Systemkomponenten adaptiert werden, so sind auch grundlegende Änderungen wie z.B. Art und Anzahl von Bearbeitungsstationen, Arbeitsplänen usw. zulässig. Grundlegend lassen sich drei Adaptionsoperationen unterscheiden: das Hinzufügen von Entitäten, das Ändern von Eigenschaften von Entitäten und das Löschen von Entitäten. Diese Operationen sind je nach Entität und deren Einbindung im Gesamtsystem durchaus sehr unterschiedlich aufwendig, wobei angenommen werden kann, dass Einfügeoperationen tendenziell einfacher zu realisieren sind als Änderungsoperationen und Löschoperationen. Diese Annahme beruht auf den starken Verknüpfungen der Entitäten untereinander und miteinander.

Einfügeoperationen bedingen per se keine Änderungen an bestehenden Systementitäten, jegliche Entität, solange in sich selbst valide, kann theoretisch hinzugefügt werden, ohne das System zu korrumpieren. So kann z.B. jederzeit eine Bearbeitungsstation eingefügt werden, ohne die Lauffähigkeit des Modells zu gefährden. Eine sinnvolle Einbindung d.h. in diesem Beispiel die Nutzung dieser Station innerhalb eines Arbeitsplans wird hierbei aber nicht sichergestellt, sondern bedingt weiteren Adaptionen.

Sollen Löschoperationen durchgeführt werden, ist zunächst sicherzustellen, dass keine negativen Effekte auf andere Entitäten, die die Lauffähigkeit des Modells gefährden, auftreten. So sind alle Referenzen auf die zu löschende Entität geeignet zu behandeln, die Behandlung kann hierbei vom einfachen Löschen oder Ändern der Referenz bis hin zum Löschen weiterer Entitäten reichen. So kann beispielsweise das Löschen einer Bearbeitungsstation zum Löschen einer gekoppelten Entität Eingangspuffer, dem Entfernen einer Referenz auf die Station in einer Liste paralleler Stationen oder zu Änderungsbedarf in Arbeitsplänen führen.

Bei Änderungsoperationen ist zwischen Änderungen unabhängiger Parameter, z.B. Verfügbarkeit, und abhängiger Parameter, z.B. Bezeichner, zu unterscheiden. Unabhängige Parameter können ähnlich den Einfügeoperationen jederzeit ohne weiteren Handlungsbedarf durchgeführt werden, bei abhängigen Parametern sind analog den Löschoperationen Referenzen zu analysieren und ggf. geeignete Maßnahmen zur Behandlung dieser zu ergreifen.

Oft sind Adaptionen eine Kombination aus einer Vielzahl einzelner Operationen, wodurch die Komplexität weiter steigt. Um diese beherrschbar zu gestalten sind zwingend Mechanismen zur Unterstützung des Anwenders bei der Adaption vorzusehen. Dabei müssen vor allem auf Ebene des Konzeptmodells Mechanismen zum Erkennen von Abhängigkeiten realisiert werden, die den Anwender zumindest auf mögliche Probleme hinweisen oder gar einzelne Operationen verhindern oder Folgeoperationen auslösen (z.B. Löschen abhängiger Entitäten), ein vollständiges automatisches Management aller auftretenden Probleme hingegen ist nicht sinnvoll möglich. Alle Prüfungen sind als Teil der Validierung und Verifikation zu sehen. Ein weiteres Teilproblem stellt in diesem Zusammenhang das sichere Erkennen von Entitäten über Modellgrenzen hinweg dar, so ist es unter Umständen außerordentlich schwierig auf Basis einer Änderung des Konzeptmodells im Simulationsmodell zwischen einer Änderungsoperation oder einer Kombination aus Lösch- und Einfügeoperationen zu unterscheiden. Beispielsweise kann das Ändern des Bezeichners einer Station im Konzeptmodell im Simulationsmodell auch als Löschen der "alten" und Anlegen einer "neuen" Station interpretiert werden. Beide alternativen Vorgehensweisen führen aber nur unter der Annahme, dass keine manuellen Ergänzungen an der betroffenen Station im Simulationsmodell vorgenommen wurden, zum gleichen Ergebnis. So kann bei Fehldeutungen z.B. manuell implementierte Logik einer Bearbeitungsstation ungewollt verloren gehen. Um dies zu verhindern sind besondere Maßnahmen bei Änderungen von Bezeichnern vorzusehen. Eine Lösungsmöglichkeit ist das Speichern einer Historie für Bezeichner, d.h. eine Liste ehemaliger Bezeichner wird dauerhaft gepflegt. Dies erleichtert auch das Nachvollziehen von Adaptionen über ggf. mehrere Adaptionsschritte hinweg.

Ein Spezialfall innerhalb der Adaption stellt die Adaption der dynamischen Abläufe im Sinn von Steuerungsregeln auf Basis von online trainierten künstlichen neuronalen Netzen, vgl. hierzu Abschnitt 3.2.2, dar.

2.3 Das Core Manufacturing Simulation Data (CMSD) Information Model als Standard für den Datenaustausch im Kontext der Simulation von Produktionssystemen

Wie bereits im Abschnitt 2.2 diskutiert, ist ein standardisiertes Datenformat bzw. Datenmodell für den gesamten Prozess der Modellgenerierung und -initialisierung von Vorteil.

In diesem Sinne ist das Core Manufacturing Simulation Data (CMSD) Information Model der vom Autor präferierte Standard zur (Daten-) Modellierung und zum Datenaustausch im Kontext der datengetriebenen Simulationsmodellgenerierung und -initialisierung sowie der Ergebnisrepräsentation von Simulationsexperimenten.

Dabei kann CMSD sowohl für die Simulation als auch für die reale Planung und Steuerung der Fertigungsabläufe genutzt werden, dies verspricht viele Vorteile, z.B. Verringerung von Redundanzen, Erhöhung der Datenintegrität, Senkung von Aufwänden bei der Simulationsmodellerstellung und Nutzung [BS2010a].

In den folgenden Abschnitten wird das CMSD Information Model detailliert vorgestellt, Stärken und Schwächen im Allgemeinen und insbesondere für die Simulationsmodellgenerierung kritisch diskutiert. Des Weiteren werden zur besseren Einordnung kurz einige weitere für den Datenaustausch, die Modellgenerierung und/oder den Modellaustausch relevante Standards abgegrenzt. Abschließend wird ein Zwischenfazit gezogen, in dem die Wahl des CMSD Information Models für weitere Betrachtungen begründet wird.

Detaillierte Beschreibungen bezüglich der vorgenommen Interpretation des Standards, die im zu entwickelnden Framework genutzt wird, erfolgt im Kapitel 3.3 Interpretation des CMSD Standards.

2.3.1 Einführung

Das Core Manufacturing Simulation Data (CMSD) Information Model ist ein offener Standard, der durch die Core Manufacturing Simulation Data (CMSD) Product Development Group (PDG) zur Standardisierung geführt wurde bzw. zurzeit geführt wird. Die PDG ist organisatorisch Teil der Simulation Interoperability Standards Organization (SISO). Besondere Beachtung galt daher im Entwicklungsprozess des Standards den durch die SISO veröffentlichten Leitsätzen, Richtlinien und Methoden [SISO2011]. Die den Standard beschreibenden Dokumente sind in Zusammenarbeit von universitären und industriellen Partnern mit Unterstützung des National Institute of Standards and Technology (NIST) entwickelt worden und stellen somit eine gemeinsame Leistung der "SISO-Community" dar.

Die Ziele des Standards [SISO2012a, S.10; LLR2006, S.3] sind:

- die Unterstützung bzw. Förderung der Entwicklung und Nutzung von Simulation im produzierenden Gewerbe,
- der erleichterte Datenaustausch zwischen Simulation und anderen Anwendungen im Kontext der Fertigung (vgl. Abbildung 14),
- das Ermöglichen bzw. die Verbesserung von Tests und Bewertungen von Software im Fertigungsbereich sowie
- die Verbesserung der Interoperabilität zwischen Simulationsanwendungen und betrieblichen Systemen der Fertigung bzw. zwischen verschiedenen Anwendungen im Kontext der Fertigung, z.B. ERP/PPS, CAP, CAD, DES u.v.m. (vgl. Abbildung 14)

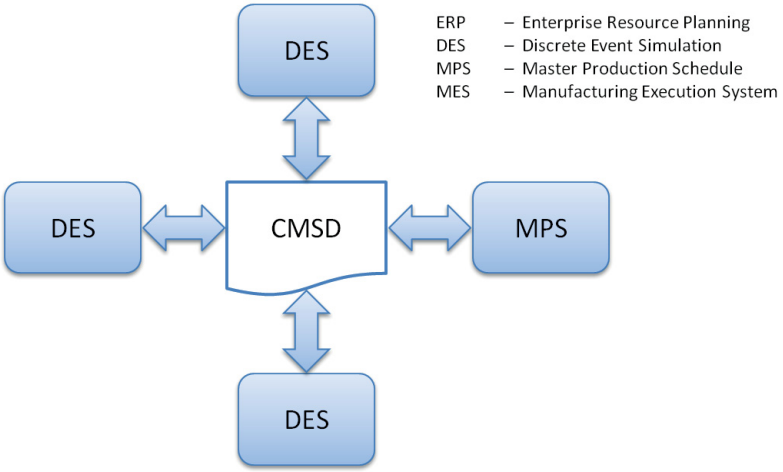


Abbildung 14: CMSD als neutrales Datenaustauschformat verschiedener Systeme der Fertigung (in Anlehnung an [Jo2007, S. 1674])

Zu bemerken ist, dass die Ziele des CMSD Standards große Gemeinsamkeiten zu den von Fowler und Rose [FR2004] veröffentlichten "Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems" oder dessen 2012er Resume "Panel on Grand Challenges for Modeling and Simulation" [Ta2012] aufweisen.

Um die Ziele zu erreichen, wird im CMSD Standard ein Datenmodell zur Beschreibung von Produktionssystemen und somit auch zur Beschreibung von Simulationsmodellen dieser Systeme, definiert.

Im Kontext dieser Arbeit wird der CMSD Standard immer in Verbindung mit Simulation genutzt. Eine Anwendung mit dem Fokus des Datenaustauschs zwischen sonstigen IT Systemen eines produzierenden Unternehmens, z.B. zwischen Planungssystemen und Steuerungssystemen ist aber ebenso denkbar. Gegenüber anderen Standards, z.B. STEP (siehe Kapitel 2.3.3.2) sind während des Designs des Standards aber explizit simulationsrelevante Faktoren mit eingeflossen, z.B. die Möglichkeit Verteilungen zu definieren [SISO2012a, S10].

2.3.2 Aufbau des CMSD Standards

Die Spezifikation des Core Manufacturing Simulation Data Information Models gliedert sich in zwei Teile, der erste Teil enthält die visuelle Darstellung des Informationsmodells in Form von Unified Modeling Language (UML) Klassen- und Paketdiagrammen [SISO20012a], im zweiten Teil werden auf Basis des ersten Teils XML Schemata der einzelnen Datenstrukturen definiert [SISO2012b]. Im Folgenden werden beide Teile soweit für das Verständnis der Arbeit nötig näher beleuchtet.

Der erste Teil des Standards mit dem Titel: "Standard for: Core Manufacturing Simulation Data – UML Model" wurde am 20. September 2010 offiziell bestätigt. Teil eins enthält die visuelle Darstellung des CMSD Information Models mittels UML Klassen- und Paketdiagrammen. Die Unified Modeling Language (UML) ist eine graphische Modellierungssprache zur Spezifikation von Software aber auch anderen Systemen, die durch die Object Management Group (OMG) entwickelt und derzeit in Version 2.3 standardisiert ist [OMG2012a, OMG2012b], für detailliertere Informationen über UML Modellierung wird auf die entsprechenden Lehrbücher , z.B. [BGB2007], verwiesen.

Ebenfalls im Standard definiert ist der grundlegende Aufbau eines konkreten CMSD Dokuments (vgl. Abbildung 15), d.h. wie einzelne Entitäten innerhalb des CMSD Dokuments auftreten bzw. kombiniert werden können. Alle Entitäten werden dabei über einen Identifier identifiziert, dabei wird streng zwischen einzigartigen (unique) und begrenzt einzigartigen (limited unique) Entitäten unterschieden, die nur im Kontext des übergeordneten Elements (Parent) einzigartig sein müssen [SISO2012a, S.17f].

Die UML Pakete werden zur thematischen Gruppierung der Klassen genutzt (vgl. Abbildung 16), die Klassen selbst bilden die Schablonen für alle definierbaren Entitäten des zu modellierenden Produktionssystems. Hierbei werden in allen Klassendiagrammen allein Attribute und Beziehungen zwischen verschiedenen Klassen bzw. Entitäten sowie deren Kardinalitäten abgebildet. Ergänzend benötigte Einschränkungen oder Erweiterungen, die nicht direkt in UML Syntax abbildbar sind, sind als Textblöcke vermerkt (vgl. Abbildung 17). Auf das in UML Klassendiagrammen mögliche Abbilden von Verhalten (Methoden) wird verzichtet, da alle Klassen als reine Daten-Container genutzt werden.

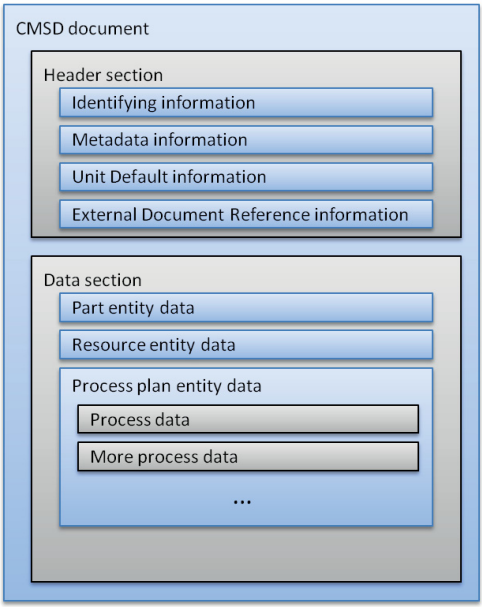


Abbildung 15: Schematischer Beispielaufbau eines CMSD Dokuments

In dem Paketdiagramm des CMSD Information Models (Abbildung 16) sind auf oberster Ebene insgesamt 6 Pakete definiert worden.

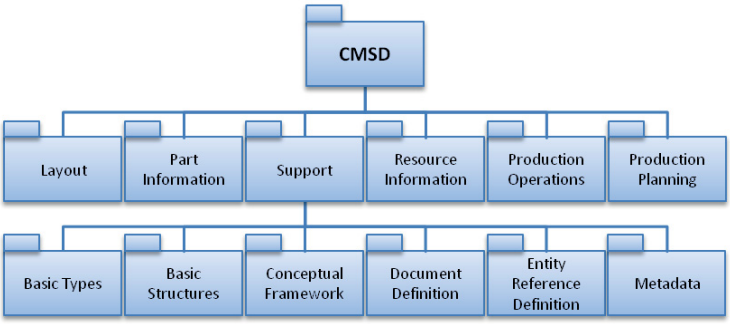


Abbildung 16: Pakete des CMSD Information Models [SISO2012a, S16]

Fünf der Pakete enthalten die Entitäten-Definitionen nach fachlich fertigungstechnischen Kriterien aufgeteilt. Abgerundet wird das CMSD Information Model durch das umfangreiche Support Paket, in welchem, aufgegliedert in 6

Subpakete, grundlegende Konstrukte wie Datentypen, Strukturen usw. definiert sind, die für die Umsetzung des CMSD Standards unabdingbar sind (siehe Tabelle 6).

Tabelle 6: Inhalte des Support Paktes

Subpaketname	Beschreibung
Basic Types	Dieses Paket enthält die Definitionen aller für die Attribute der Klassen erlaubten Datentypen. Das Spektrum reicht von simplen Typen wie z.B. String und Integer, bis hin zu Auflistungen von erlaubten Werten für beispielsweise Maßeinheiten [SISO2012a, S.52-64]
Basic Structures	Dieses Paket enthält Definitionen einfacher Strukturen, welche in komplexeren Strukturen oder Klassen verwendet werden, beispielsweise werden die möglichen Strukturen der Properties definiert [SISO2012a, S21-52]
Conceptual Framework	In diesem Paket wird beschrieben, wie einzelne Entitäten zueinander in Beziehung stehen können. Es wird definiert, ob und wann eine Entität begrenzt einzigartig oder einzigartig ist. Ebenfalls ist hier die Basisklasse "Entity", von der sämtliche anderen Klassen von Entitäten abgeleitet sind, enthalten [SISO2012a, S67-70].
Document Definition	In diesem Paket ist die Grundstruktur für CMSD Dokumente definiert (vgl. Abbildung 15) [SISO2012a, S.71-76]
Entity Reference Definition	Dieses Paket enthält die Definition aller erlaubten Referenzen, die genutzt werden, um Verweise von einer auf andere Entitäten zu realisieren, dabei sind sowohl einfache als auch komplexe Verweise möglich [SISO2012a, S.77-86].
Metadata	In diesem Paket wird beschrieben, wie eigene Properties definiert werden können [SISO2012a, S.107-110].

Das "**Support**" Paket selbst enthält keinerlei Klassen, die direkt Entitäten beschreiben, alle (Produktions-) System Entitäten sind allein in den 5 im Folgenden kurz beschriebenen Paketen definiert. Eine 1:1 Umsetzung auf die in Abschnitt 2.1.6 diskutierten Eingangsdaten der Simulation (vgl. Abbildung 5) nach VDI3633 Blatt 1 [VDI3633-1] ist indes nicht möglich, da unterschiedliche Strukturierungsmerkmale genutzt wurden.

Das "**Layout**" Paket [SISO2012a; S.87-106; RL2008], als ein erst spät in den Standard aufgenommener Teil, bietet die Möglichkeit, Informationen über die Anordnung von Objekten in einem Fertigungs- bzw. Hallenlayout zu modellieren, sowie deren Abmessung und Form zu beschreiben. Ziel ist es, eine Visualisierung der Fertigung zu ermöglichen bzw. Informationen abzubilden, die für logistische Analysen des Fertigungslayouts benötigt werden. Die Anwendung als Datenaustauschformat im Sinne von Zeichnungsdaten, z.B. für CAD-Systeme wird explizit nicht angestrebt.

Innerhalb des "**Part Information**" Paketes [SISO2012A; S.11-118] werden sämtliche Aspekte, die Rohstoffe, Teilkomponenten und Endprodukte betreffen, subsumiert. Dabei sind sowohl konkrete Ausprägungen ("Parts") als auch Typbeschreibungen möglich ("Part Type"). Ebenfalls in diesen Paket hinterlegt sind Klassen für das Abbilden von Stücklisten und Inventargegenständen.

Die im "**Resource Information**" Paket [SISO2012A; S. 143-152] definierten Klassen ermöglichen die Abbildung aller für die Fertigung relevanten Ressourcen (vgl. Abbildung 17), d.h. sowohl Arbeitsplätze, Maschinen, Fördertechnik usw. als auch Werker. Ebenso sind für die Abbildung relevante Informationen wie Fähigkeiten von Werkern ("Skills"), Rüstmatrizen ("Setups") und das Bilden von Maschinengruppen möglich.

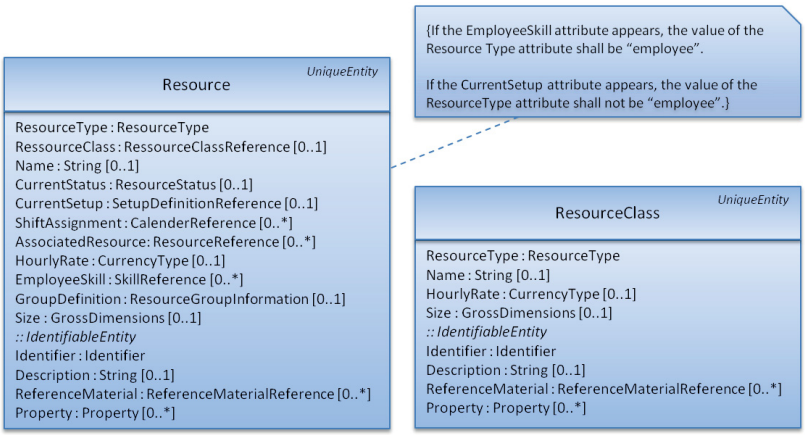


Abbildung 17: Eine CMSD Klassennotation am Beispiel der Resource und ResourceClass Klasse [SISO2012a, S.143]

Im "**Production Operations**" Paket [SISO2012A; S. 119-130] sind zunächst die Klassen gekapselt, die Kunden- ("Order") bzw. Fertigungsaufträge ("Job"), inklusive der geplanten und aktuellen Zustände und Aufwände, repräsentieren. Ebenso in diesem Paket sind Klassen zur Beschreibung von Zeitplänen ("Schedule") definiert.

Das "**Production Planning**" Paket [SISO2012A; S. 131-142] kapselt schließlich den für die Produktion unabdingbaren Aspekt der Prozess- bzw. Arbeitspläne, Wartungspläne sowie sämtliche Informationen bzgl. Arbeitszeiten, d.h. Schicht-, Pausen- und Urlaubskalender.

Trotz der Gruppierung in verschiedene Pakete ist erst durch die Kombination verschiedener Klassen eine umfassende und sinnvolle Systemabbildung möglich, z.B. können Informationen aus dem Part Information Paket über Rohstoffe und Endprodukte, ebenso wie Informationen aus dem Ressource Paket über Maschinen, deren Rüstzustände und Werker im Production Planning Paket zur Definition eines Arbeitsplanes genutzt werden.

CMSD bietet neben den bereits ausdefinierten Attributen der Klassen die Möglichkeit, den Standard um eigene Attribute zu erweitern. Die (Nutzer-) Erweiterungen werden über das sogenannte Property Konzept realisiert [SISO2012a, S.21, 107; LLR2006, S3f]. Properties sind begrenzt frei belegbare Attribute, die in den meisten Klassen vorhanden sind, z.B. der Klasse Ressource (vgl. Abbildung 17). Realisiert werden können drei unterschiedlich mächtige Property Arten (vgl. Abbildung 18), die "SimpleProperty", die allein einen Textwert annehmen kann, die "StochasticProperty", welche eine Verteilungsfunktion abbildet und die "ReferenceProperty", welche Verweise auf andere Entitäten ermöglicht.

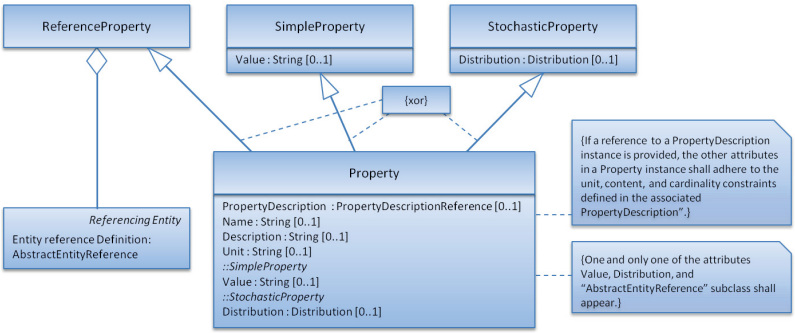


Abbildung 18: Die CMSD Property Klassen [SISO2012a, S.21]

Der zweite Teil mit dem Titel: "Standard for: Core Manufacturing Simulation Data – XML Representation" wurde im Oktober 2012 offiziell als Standard veröffentlicht [SISO2012b].

Nachdem der erste Teil des Standards mittels UML alle Entitäten und Hilfsklassen definiert hat, wird darauf aufbauend im Teil zwei die für die tatsächliche Nutzung als Datenaustauschformat wichtige Beschreibung mittels einer XML Schema Sprache vorgenommen, bzgl. der Entitäten, Hilfsklassen und deren Attribute sind beide Teile inhaltlich völlig identisch.

XML ist wie bereits erwähnt eine verbreitete Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien, beispielsweise zum

Datenaustausch zwischen verschiedenen IT-Systemen [W3C2012, SISO2012b, S.12]. Zur Definition domänenspezifischer XML basierter Sprachen werden sogenannte XML Schemasprachen genutzt. Mittels dieser kann definiert werden, welche Elemente, welche Struktur/Anordnung und ggf. welche Wertebereiche innerhalb eines XML-Dokumentes erlaubt sind [Ho2009].

Im Entwicklungsprozess des 2. Teils des Standards wurden auf Grund der hohen Komplexität und vielfältigen Verknüpfungen der einzelnen Entitäten, wie in Teil eins des Standards beschrieben, Regular Language for XML Next Generation (RELAX NG) und Schematron als Schemasprache gewählt [SISO2012b, S.12].

RELAX NG ist eine durch die ISO unter der Nummer 19757-2 standardisierte Schemasprache zur Beschreibung von Strukturen in XML Dokumenten [ISO19757-2], für Details zu RELAX NG wird auf entsprechende Literatur, z.B. [Va2004] verwiesen. Schematron ist eine ebenfalls durch die ISO veröffentliche regelbasierte Schemasprache, die als Ergänzung zu strukturbeschreibenden Sprachen genutzt werden kann [ISO19757-3]. Ziel von Schematron ist es, ergänzend zu grammatikalisch strukturellen Regeln, in diesen Fall mittels RELAX NG, auch inhaltliche Regeln abzubilden, die sich aus dem aktuellen sachlichen Zusammenhang ergeben, für Details zu Schematron wird ebenfalls auf entsprechende Literatur, z.B. [HMK2011] verwiesen.

Beispielhaft ist in Abbildung 19 die RELAX NG Beschreibung der Ressource Klasse (vgl. UML Darstellung in Abbildung 17) abgebildet, in der sämtliche Attribute, deren Datentypen und Häufigkeiten definiert sind.

```
<rng:define name="Resource">
        <rng:interleave>
                <rng:ref name="IdentifiableEntity"/>
                <rng:element name="ResourceType">
                        <rng:ref name="ResourceType"/>
                </rng:element>
                <rng:optional>
                        <rng:element name="ResourceClass">
                                <rng:ref name="ResourceClassReference"/>
                        </rng:element>
                </rng:optional>
                <rng:optional>
                        <rng:element name="Name">
                                <rng:ref name="String"/>
                        </rng:element>
                </rng:optional>
                <rng:optional>
                        <rng:element name="CurrentStatus">
                                <rng:ref name="ResourceStatus"/>
                        </rng:element>
                </rng:optional>
                <rng:optional>
                        <rng:element name="CurrentSetup">
                                <rng:ref name="SetupDefinitionReference"/>
                        </rng:element>
                </rng:optional>
                <rng:zeroOrMore>
                        <rng:element name="ShiftAssignment">
                                <rng:ref name="CalendarReference"/>
                        </rng:element>
                </rng:zeroOrMore>
                <rng:zeroOrMore>
                        <rng:element name="AssociatedResource">
                                <rng:ref name="ResourceReference"/>
                        </rng:element>
                </rng:zeroOrMore>
                <rng:optional>
                        <rng:element name="HourlyRate">
                                <rng:ref name="CurrencyType"/>
                        </rng:element>
                </rng:optional>
                <rng:zeroOrMore>
                        <rng:element name="EmployeeSkill">
                                <rng:ref name="SkillReference"/>
                        </rng:element>
                </rng:zeroOrMore>
                <rng:optional>
                        <rng:element name="GroupDefinition">
                                <rng:ref name="ResourceGroupInformation"/>
                        </rng:element>
                </rng:optional>
                <rng:optional>
                        <rng:element name="Size">
                                <rng:ref name="GrossDimensions"/>
                        </rng:element>
                </rng:optional>
        </rng:interleave>
</rng:define>
```

Abbildung 19: Auszug aus der CMSD RelaxNG Beschreibung - Beispiel Resource Klasse [SISO2012b, S.79-81]

Die im UML Klassendiagramm textuell beschriebene sachlich, fachliche Einschränkung, z.B. dass Mitarbeiter ("Employee") nur Fähigkeiten ("Skills") und keine Rüstmatrix aufweisen, ist in RELAX NG nicht abbildbar. Die Abbildungen dieser Einschränkung wird erst durch die Einbeziehung der Schematron Regeln (Abbildung 20) möglich.

```
<sch:pattern xmlns:xs="http://www.w.org//XMLSchema"
    xmlns:a="http://relaxng.org/ns/compatibility/annotations/."
    name="Resource.EmployeeSkill check">
    <sch:rule context="//DataSection/Resource[EmployeeSkill]">
        <sch:assert test="ResourceType = 'employee'">
          If EmployeeSkill is present  in a  Resource element, the ResourceType must be 'employee'
        </sch:assert>
    </sch:rule>
</sch:pattern>
<sch:pattern xmlns:xs="http://www.w.org//XMLSchema"
    xmlns:a="http://relaxng.org/ns/compatibility/annotations/."
    name="Resource.CurrentSetup check">
    <sch:rule context="//DataSection/Resource[CurrentSetup]">
        <sch:assert test="ResourceType != 'employee'">
            If an CurrentSetup  element is present in a Resource element, the ResourceType must not
            be 'employee'
        </sch:assert>
    </sch:rule>
</sch:pattern>
```

Abbildung 20: Auszug aus der CMSD Schematron Beschreibung - Beispiel Resource Klasse [SISO2012b, S.121]

Wie bereits beschrieben, wird durch die Abbildung als Schema zum einen eine automatische Validierung von CMSD XML Dateien auf Standardkonformität ermöglicht, zum anderen erlauben viele Programmierumgebungen heute das Einlesen von Schemata zum automatischen Generieren von Klassen, zum Erzeugen und Manipulieren von schemakonformen XML Dateien.

2.3.3 Weitere relevante Standards

Das CMSD Information Model ist bei Weitem nicht der einzige Datenstandard, der in bisherigen Arbeiten zur Modellgenerierung und Initialisierung genutzt wurde. Zusätzlich kommen weitere Datenformate /-Standards, aus dem Bereich der Modellierung von Systemen und Geschäftsprozessen sowie dem gesamten Bereich der Produktdatenabbildung in Betracht. Im Folgenden wird eine kleine Auswahl der vom Autor untersuchten Standards hinsichtlich ihrer Eignung hin vorgestellt. Die Auswahl wurde in Bezug auf die Anwendungshäufigkeit in Fachpublikationen der letzten Jahre und der Verbreitung in der Praxis bzw. Implementierung in Simulationswerkzeugen getroffen.

2.3.3.1 Simulation Data Exchange (SDX)

Der Simulation Data Exchange, kurz SDX, Standard ist ursprünglich als allgemeingültiges Datenformat zum Austausch von Daten zur Modellgenerierung und 3D Visualisierung aus CAD Systemen heraus entwickelt worden. Der Ansatz basiert dabei auf der Anreicherung von Objekten im CAD System, sogenannten "smart objects", um simulationsrelevante Informationen. Aktuell sind 27 Klassen von "smart objects" definiert, z.B. Machines oder Conveyers. Je Klasse sind sogenannte SDX Properties hinterlegt, die zu pflegen sind, so zum Beispiel die Felder Name, Typ, Rüstzustände uvm. für die Maschinenklasse oder Name und Geschwindigkeit usw. für die Fördererklasse. In SDX lassen sich darüber hinaus Materialflüsse/Arbeitspläne rudimentär abbilden, die Abbildung von Lastdaten und sonstigen Initialisierungsdaten hingegen ist nicht vorgesehen. [SM2001, S. 1473-1474; Mo2006, S. 30-39].

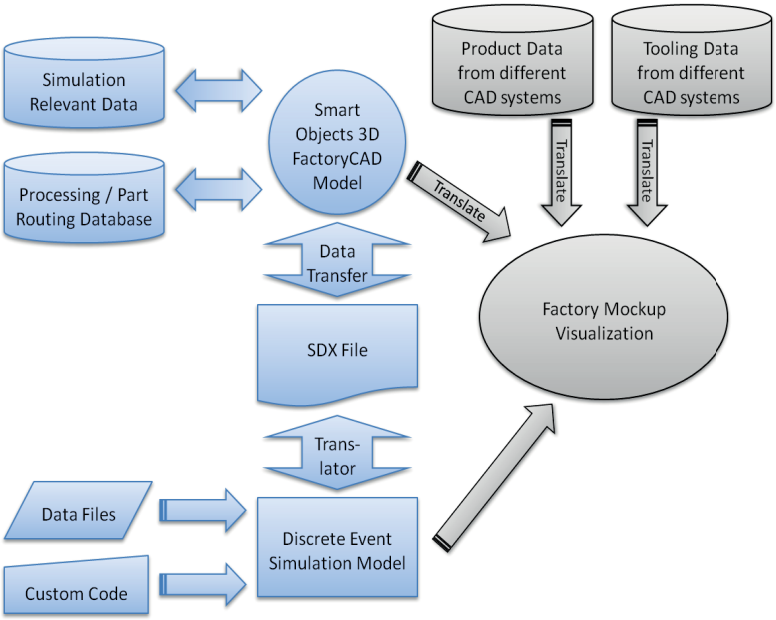


Abbildung 21: SDX generic Architecture [Mo2006, S.31]

Der prinzipielle Ablauf einer Simulationsstudie unter Nutzung von SDX kann grob in folgende Schritte aufgeteilt werden (vgl. Abbildung 21):

1. Erzeugen eines 3D Layouts (in factoryCAD[12]),
2. Ergänzen der simulationsrelevanten Daten zu den Fabrikobjekten (smart objects),
3. Definition der Materialflüsse/Arbeitspläne mittels eines externen Editors "SDX Route Editor",
4. Generierung des SDX Datenfiles als Text in der ASCII Codierung, welches die Beschreibung der Fertigungs- und Layoutdaten beinhaltet,
5. Überführung des SDX Files in eine kompatible Simulationsumgebung (z.B. Plant Simualtion)
6. Durchführung des/der Simulationsläufe bzw. -experimente [He2001, S.1470-1471; Mo2006, S. 30-32].

Insgesamt ist SDX durchaus ein zumindest für den Anwendungsfall der planungsbegleitenden Simulation vor allem unter dem Aspekt der Layoutplanung geeigneter Standard. Trotz allem bestehen erhebliche Einschränkungen sowie Kompatibilitätsprobleme [Mc2003]. So unterstützen zwar eine kleine Anzahl von Simulationswerkzeugen[13] SDX, eine Nutzung von SDX ohne den Einsatz des Werkzeugs factoryCAD ist hingegen nicht sinnvoll möglich. Dies schränkt den prinzipiellen Anwendungskontext stark ein und bindet den Anwender an ein bestimmtes Produkt. Zudem ist SDX nur ein Industriestandard der nicht allgemeinverfügbar und durch den Rechteinhaber jederzeit veränderbar ist. Zudem sind keinerlei Erweiterungsmöglichkeiten vorgesehen. Ebenso sind Aspekte bzgl. Last - und Initialisierungsdaten, Strategien und Werker und viele Dinge mehr in SDX nicht abbildbar.

2.3.3.2 Standard for The Exchange of Product model data (STEP)

Eine weitere Klasse von Standards, die prinzipiell für Modellgenerierung in Betracht kommt, stellen Standards aus dem Bereich des Produktdatenmanagements dar. Gegenüber Standards, die nur reine Geometrie (z.B. IGES) abbilden, werden in diesen meist auch nicht physische funktionale Aspekte des Produkts abgebildet. Der verbreitetste und auch wie in Abschnitt 2.2.1 diskutiert, bereits in Ansätzen zur automatischen Modellgenerierung eingesetzte Standard ist der "Standard for The Exchange of Product model data" (STEP), der in der Normenreihe ISO 10303 umfangreich beschrieben ist [ISO 10303-1; Sp1995, S.60-63; AT2000].

[12] factoryCAD ist ein CAD Werkzeug zur Erstellung von Fabriklayouts der Firma Siemens PLM Software; http://www.plm.automation.siemens.com/de_de/products/tecnomatix/plant_design/factorycad
[13] z.B. Simul8, Plant Simulation

Ziel von STEP ist es, über den gesamten Produktlebenszyklus hinweg möglichst alle anfallenden Produktdaten darzustellen und den Datenaustausch zwischen unterschiedlichen CAx-Systemen (CAD, CAM, CAE usw.) zu ermöglichen. Dazu werden Informationen aus den Bereichen Konstruktion, Berechnung, Fertigung/Montage, Qualitätssicherung, Wartung und Betrieb abgebildet. In STEP wird dazu ein Gesamtmodell herangezogen, das aus diversen verknüpften Partialmodellen, die je in eigenen Teilnormen den Applikationsprotokollen (AP) definiert sind [ISO 10303-1; AT2000].

Der Fokus von STEP liegt klar auf dem Produkt selbst, weniger auf dem Fertigungssystem. Zwar besteht mit STEP die Möglichkeit, z.B. Informationen über die hierarchische Struktur, technische Arbeitspläne usw. weiterzugeben, viele simulationsrelevante Daten lassen sich aber nicht oder zumindest nur auf Umwegen in STEP abbilden. Dazu zählen Daten bzgl. der Systemlast ebenso wie beispielsweise Daten bzgl. der Werker, Strategien, Schichten usw. Eine weitere Schwäche dieser produktorientierten Sichtweise ist bei Fertigungssystemen zu sehen, die mehr als ein Produkt produzieren bzw. hohe technische Freiheitsgrade besitzen. Hier erhöht sich der Aufwand für die Datenmodellierung aber auch die Nacharbeit deutlich.

2.3.3.3 Specification and Description Language (SDL)

Neben den "reinen" Datenstandards sind auch diverse Modellierungssprachen zum Datenaustausch denkbar. Dabei sind aber nur Modellierungssprachen denkbar, die neben der grafischen Repräsentation auch über Möglichkeiten der strukturierten Speicherung der Daten, z.B. in XML verfügen. Ein Vertreter dieser Gruppe von Sprachen ist die Specification and Description Language (SDL), welche zur Modellgenerierung erfolgreich eingesetzt wird, so nutzt Fonseca i Casas SDL um DES Modelle zu erstellen [FC2010]. Eine Anwendung im Kontext der Produktion ist bisher nicht erfolgt.

SDL wurde als Modellierungssprache von der Internationalen Fernmeldeunion in den Standards Z.100ff definiert und ist im Telekommunikationsbereich durchaus verbreitet. Anwendung ist die Modellierung (verteilter Echtzeit-) Systeme mittels erweiterter Zustandsautomaten (Prozesse). Die über Kanäle verbundenen Prozesse können Signale austauschen und sich so gegenseitig beeinflussen. Die Abbildung kann entweder als Text (Textual Phrase Representation) und/oder in graphischer Form (Graphic Representation) erfolgen [ITU Z.100].

SDL ist zwar bewiesenermaßen geeignet, die automatische Generierung auch von diskret ereignisorientierten Simulationsmodellen zu unterstützen, der Aufwand der Modellierung von Produktionssystemen mittels der Zustandsautomaten ist aber tendenziell sehr hoch. Des Weiteren ist die Verbreitung außerhalb der Telekommunikationsbranche als sehr gering einzuschätzen. Im Kontext der in dieser

Arbeit angestrebten datengetriebenen Modellgenerierung ist zudem die grafische Repräsentation weitestgehend überflüssig. Die benötigte Erzeugung der proprietären nicht XML basierten textuellen Darstellung aus ggf. verschiedenen Quellsystemen ist mit erheblichem Aufwand verbunden.

2.3.3.4 SysML

Eine zweite grafische Modellierungssprache, die zur Modellgenerierung genutzt wird, ist die Systems Modeling Language (SysML). SysML ist ein für den Bereich des Systems Engineering durch die Object Management Group (OMG) gepflegter Standard, der auf UML aufbaut. Ziel des Standards ist die Unterstützung der Analyse, des Designs und des Tests von komplexen Systemen. Neben bestehenden Diagrammtypen (Sequenz-, Use Case-, Zustands- und Paketdiagramm) der UML bzw. Erweiterungen dieser (Aktivitäts-, Block Definition und Internal Block Diagramm) werden in SysML auch eigene Diagrammtypen (Requirement und Parametric Diagram) definiert [OMG2012c].

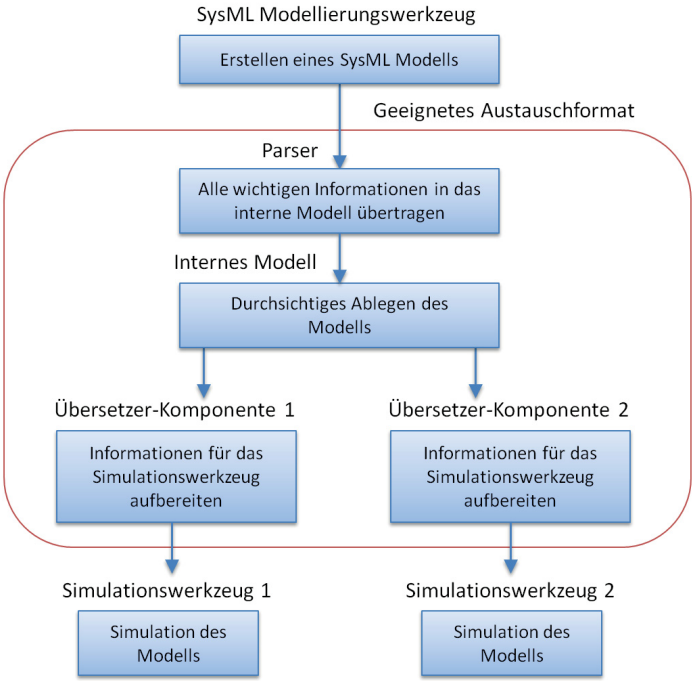


Abbildung 22: Systemarchitektur SysML Modellgenerierung [SR2010, S. 455]

Verschiedene Autoren nutzen SysML im Kontext der automatischen Modellgenerierung, so wird dieser Ansatz sowohl von Schönherr und Rose [SR2009; SR2010; SR2011] als

auch von McGinnis und Kollegen [HRM2007; MU2009; TM2011] seit Jahren verfolgt. Die Autoren bringen jedoch teilweise unterschiedliche SysML Diagramme zum Einsatz. SysML dient aber in allen Ansätzen als grafische Modellierungssprache (vgl. Abbildung 22). Das direkte Koppeln von Drittsystemen ist nicht vorgesehen.

SysML ist zwar mittels XML Metadata Interchange (XMI) serialisierbar, ist aber kein Datenstandard. Datengetriebene Modellgenerierung ist somit ebenso wie bei SDL nur über Umwege möglich. Positiv zu bemerken ist, dass SysML eine durchaus bessere Unterstützung der Modellierung von Produktionssystemen als SDL bietet. Negativ kann sich auswirken, dass für jede Modellierung mehrere Diagramme zu nutzen sind, die aber aufeinander abgestimmt sein müssen.

2.3.3.5 Sonstige

Zusätzlich zu den genannten Standards sind im Kontext der Modellgenerierung durchaus noch eine Vielzahl weiterer Standards, zumindest für Teilbereiche denkbar und wurden im Rahmen der Dissertation untersucht. Auf eine detaillierte Betrachtung soll aber verzichtet werden.

So können ähnlich wie SysML auch Sprachen zur Geschäftsprozessmodellierung genutzt werden, dies wird unter anderen von Kloos in seiner Dissertation diskutiert [Kl2013].

Einen weiteren, recht jungen Standard stellt AutomationML [Dr2010; GD2013] dar, dieser Ansatz zielt auf die umfassende Abbildung aller Daten und deren Austausch im Kontext der Digitalen Fabrik ab, explizit wird auch der Datenaustausch mit der Materialflusssimulation als mögliches Anwendungsgebiet dargestellt. Die Besonderheit des Standards liegt darin, dass kaum eigene Formate definiert werden, sondern dem Ansatz gefolgt wird, existierende Standards zu kombinieren. So ist auch eine Nutzung von z.B. CMSD in AutomationML denkbar.

Neben Modellierungssprachen können auch Ontologien zur Modellgenerierung genutzt werden, in diesen Zusammenhang ist die an der University of Georgia entwickelte Discrete Event Modeling Ontology (DeMO) [Si2009] zu erwähnen sowie die Arbeit von Lacy [La2006], der die Web Ontology Language (OWL) nutzt.

Zumindest für eine teilautomatisierte Modellgenerierung sind auch CAD Datenaustauschformate, wie beispielsweise die Initial Graphics Exchange Specification (IGES) vorstellbar, aber schon Selke bemerkt in seiner Dissertation, dass: "die Interpretation von CAD–Layoutdaten [...] nur dann erfolgreich sein [kann], wenn starke Abhängigkeiten zwischen Produktionsorganisation und Hallentopologie gegeben ist. Das wird insbesondere bei starr verketteten Anlagen der Fall sein, weniger jedoch bei flexibleren Organisationsformen (z.B. Werkstatt– oder Inselfertigung)" [Se2005, S. 27].

2.3.4 Fazit

Die Analyse des CMSD Information Models hat gezeigt, dass es die selbst gestellten Ziele erfüllt und durchaus gut geeignet zur Modellierung von Produktionssystemen erscheint. Es kann zusammenfassend gesagt werden, dass mit Hilfe von CMSD durchaus Fertigungssysteme unterschiedlichster Art sowie auch verschiedene Abstraktionslevel abgebildet werden können.

Auf Grund der relativen Neuheit des Standards ist die Verbreitung in Forschung und industrieller Praxis sowie unter den Herstellern von Simulationssoftware noch sehr gering. Erste Beispielprojekte bestätigen aber die vom Autor getroffenen und im Kapitel 3 weiter detaillierten Aussagen zur Eignung des Standards weitestgehend, so wurde im Rahmen eines schwedischen Forschungsprojektes (FACTS) CMSD als Datenaustauschformat erfolgreich eingesetzt [Jo2007, Le2008]. Als Testszenario diente ein Ausschnitt einer Automobilmontage bzw. Lackiererei der Volvo Car Corporation in Göteborg Schweden. Zu bemerken ist, dass im Rahmen des Projektes eine sehr frühe Fassung des Standards zum Einsatz kam, in dem z.B. noch keinerlei Layout-Informationen hinterlegbar waren sowie das gewählt Beispiel keine Schlüsse auf die Nutzung außerhalb der Automobilindustrie und der dort vorherrschenden Linienfertigung zulässt.

Bei allen positiven Aspekten des CMSD Standards ist zu bemerken, dass er gerade durch die aktuell noch geringe Verbreitung wenig hinsichtlich Praxiseignung validiert ist. Somit ist für zukünftige Versionen noch deutliches Verbesserungspotential zu erkennen. Darüber hinaus ist es durchaus positiv zu bewerten, dass der Standard verschiedene Sichten des Nutzers bei der Modellierung unterstützt und unterschiedliche Abstraktionsebenen, je nach Aufgabenstellung, zulässt, aber gerade diese Freiheit führt dazu, dass CMSD interpretierbar ist, d.h. Sachverhalte können ggf. auf sehr unterschiedliche Weise abgebildet werden. So kommen oft verschiedene Entitäten bzw. Attribute für ein und dieselbe zu modellierende Systemeigenschaft in Betracht. Ein weiterer Fakt, der die Variabilität des Standards erhöht, aber auch eine Nutzung gerade über Systemgrenzen hinaus erschwert, ist, dass außer dem Identifier (Teil aller Identifiable Entities) der einzelnen Elemente kaum Pflichtfelder definiert sind.

Eine weitere Grenze des Standards ist vor allem bei der detaillierten Abbildung von Verhalten zu sehen [BSS2010a, S.467]. Das ist jedoch ein generelles Problem aller diskutierten Standards. Dieser für die Simulation sehr bedeutende Punkt der Abbildung dynamischen Verhaltens wird in Abschnitt 3.2 gesondert thematisiert, die Integration eines auf "micro functions" basierenden Lösungsansatzes in CMSD wird in Abschnitt 3.3 aufgegriffen.

Eine anerkannte Referenzimplementierung für zumindest die häufigsten abzubildenden Sachverhalte von typischen Produktionssystemen in CMSD ist nicht verfügbar. Das Fehlen dieses gemeinsamen Verständnisses hemmt, nach Meinung des Autors, die Nutzung des Standards immens. So ist für eine Kopplung von betrieblichen IT-Systemen (z.B. ERP-Systemen) untereinander und/oder mit Simulationswerkzeugen eine einheitliche Nutzung von Attributen sowie Konventionen bzgl. der Nutzung von zusätzlichen Properties zwingend erforderlich, um den Entwicklungsaufwand zu rechtfertigen und breite Interoperabilität zu gewährleisten.

Im folgenden Kapitel wird daher unter anderem ein Vorschlag zur Interpretation, zu Modellierungspattern, zu Pflichtfeldern und zu Erweiterungen in Form von Properties beschrieben. Dieser Interpretationsvorschlag soll dabei möglichst breit ausfallen und somit viele Produktionssysteme abdecken können.

2.4 Zusammenfassung der resultierenden Forschungsfragen

Auf Basis der vorangegangenen Betrachtungen ergeben sich folgende primäre Forschungsfragen, denen im Rahmen dieses Dissertationsprojektes nachgegangen wird:

1. Wie muss ein Modell bzw. eine Modellierungsmethodik gestaltet sein, um eine sinnvolle automatische Modellgenerierung sowie Modelladaption zu ermöglichen?
2. Wie kann die Qualität eines automatisch generierten Simulationsmodells einer Fertigung auch langfristig und lebenszyklusphasenübergreifend erhalten bzw. laufend verbessert werden?
3. Welche Daten werden benötigt, wie lassen Sie sich abbilden? Welche Datenformate können wie genutzt werden?

Neben den primären Forschungsfragen sind einige sekundäre Forschungsfragen abzuleiten, die nur auszugsweise und nicht abschließend im Rahmen dieser Arbeit beantwortet werden sollen:

a. Wie kann dynamisches Verhalten abgebildet werden, so dass eine Nutzung in der Simulation möglich wird?
b. Wie kann Modellqualität beurteilt werden? Welche automatisierbaren Mechanismen der Validierung und Verifikation sind geeignet?
c. Wie kann sichergestellt werden, dass sich das Modell an Veränderungen der Realwelt zeitnah anpasst?
d. Welche Informationssysteme sind als Datenquellen für die Modellgenerierung/-adaption geeignet? Welche Bedingungen werden dabei an die Informationssysteme gestellt? Wie kann eine Integration der Informationssysteme und der Simulation erfolgen?

3 CMSD basiertes Framework zur automatischen Simulationsmodellgenerierung, -adaption und -validierung

Im Folgenden wird ein umfangreiches Framework zur Simulationsmodellgenerierung, -nutzung, -adaption und -validierung beschrieben, sowie ein angepasstes Vorgehensmodell und Integrationsansätze in die betriebliche IT-Infrastruktur vorgestellt. Einen wesentlichen Teil dieses Kapitels bildet die Interpretation des CMSD Standards als die Kernkomponente des Ansatzes. Ebenso wird die bereits in Abschnitt 2.2 skizzierte Problematik der Modellierung dynamischen Verhaltens bzw. des Handhabens dynamischer Aspekte in der Modellgenerierung aufgegriffen, dazu werden kurz bestehende und im Framework implementierte Ansätze betrachtet sowie ein eigener auf Künstlichen Neuronalen Netzen (KNN) beruhender Ansatz vorgestellt.

Zunächst wird das Framework im Kontext der Modellgenerierungsansätze eingeordnet. Der hier vorgestellte CMSD basierte Ansatz ist klar als datengetriebene Modellgenerierung einzuordnen. Nach dem Klassifizierungsschema von Eckardt [Ec2002, S.39] handelt es sich um einen hybrid-wissensbasierten Ansatz, da neben strukturbasierten Aspekten, d.h. der Nutzung der Layoutinformationen, auch Parameterdaten eine entscheidende Rolle spielen. Des Weiteren kommen z.B. bei der Datenvalidierung, Abbildung dynamischen Verhaltens usw. Methoden der künstlichen Intelligenz, z.B. KNN oder Entscheidungsbäume zum Einsatz.

In Tabelle 7 ist weiterhin eine Einordnung nach den bereits in Abschnitt 2.2 in Form eines morphologischen Kastens aufgeführten Klassifizierungsmerkmalen vorgenommen. Dabei wurden alle den hier vorgestellten Ansatz am besten kennzeichnende Ausprägungen markiert (grün hinterlegte Zelle). Zusammenfassend ist zu erkennen, dass gegenüber bestehenden Ansätzen ein deutlich breiterer Anwendungsbereich gegeben ist. So ist weder eine strikte Einordnung in die Planungs-, Inbetriebnahme- oder Betriebsphase nötig, noch die Fokussierung auf einzelne Gewerke oder Teilaspekte (z.B. nur Fördertechnik) der Fertigung. Des Weiteren wird durch die angestrebte Vollautomatisierung, Unterstützung aller Simulationsphasen und den generischen Modellaufbau unter Nutzung von Datenstandards eine Anwendung durch nicht Simulationsexperten ermöglicht bzw. erleichtert. Zu bemerken ist, dass auf Simulationsexperten keinesfalls komplett verzichtet werden kann, allein der Tätigkeitsschwerpunkt verschiebt sich durch den hier vorgestellten Ansatz (vgl. Abschnitt 3.1).

Tabelle 7: Klassifikation des CMSD basierten Ansatzes (in Anlehnung an [SBM2010])

Kriterium	*Ausprägung*				
Einsatzfall der generierten Simulations-modelle	planungsbegleitend		betriebsbegleitend	hybrid	
	strategisch (z.B. Standort)	taktisch (z.B. Layout, Puffergrößen)			
Fertigungstyp	Job-Shop	Flow-Shop	Open-Shop		
Fokussiertes Gewerk	Montage	Lackiererei	Logistik	Förder-technik	unabhängig
Nutzergruppe	Fachabteilung	Simulationsexperte	Jeder		
Grad der Automatisierung der Modellerstellung	keine	teilautomatisch		voll automatisch	
		Struktur	Verhalten	beides	
Ansatz der Modellerstellung	Standardi-sierte Eingabe-formulare	Dialoggeführt (Wizzard o.ä.)	Referenz-modelle	Direkter generischer Modellaufbau	
				eine Daten-quelle	alle rele-vanten Systeme
Unterstützte Phasen der Simulationsstudie	Modellerstellung	Verifikation und Validierung	Experiment / Initialisierung	mehrere oder alle	
Technische Umsetzung der Schnittstelle	textbasiert (z.B. *.csv)	tabellenbasiert (z.B. Excel)	XML (z.B. CMSD)	Datenbank (z.B. SQL)	
Art der fachlichen Schnittstelle	layoutbasiert (z.B. SDX)	produktbasiert (z.B. STEP)	prozessbasiert	sonstige / hybrid	
Wieder-verwendung des Modells	keine		mehrmalig		
	keine weiteren Fragestell-ungen	nicht wirt-schaftlich	manuelle Nachbe-arbeitung	Neupara-metrisierung	kontinu-ierliche Anpassung

Wie bereits dargelegt, ist die Automatisierung aller im Rahmen der Simulation auftretenden Teilprozesse durchaus sinnvoll. So kann durch die Automatisierung unter anderen die Nutzbarkeit der Simulation für Nicht-Experten verbessert, der Zeitbedarf und somit die Kosten einer Studie gesenkt sowie die Qualität der Modelle erhöht, deren Vergleichbarkeit erleichtert und nicht zuletzt die Wiederholbarkeit (Reproduktion) von (Modellierungs-) Ergebnissen ermöglicht werden.

Einige der beschriebenen positiven Effekte treten mehr oder weniger per se bei jeglicher automatisierter Modellgenerierung ein, andere bedürfen begleitende Maßnahmen, wie beispielsweise entsprechend gestaltete Validierung und Verifikation oder Vorgehensweisen für die Simulation und Generatorimplementierung. Die entsprechenden Anforderungen werden im Folgenden schrittweise thematisiert. So ist festzuhalten, dass eine einzelne technische oder organisatorische Komponente allein keine den Zielen der Arbeit adäquate Lösung liefert, erst das Zusammenspiel verschiedener Werkzeuge und die konsequente Einbettung in Vorgehen und Organisation ermöglichen die Zielerreichung.

3.1 Angepasstes Vorgehensmodel zur automatischen Simulationsmodellgenerierung, -adaption und -validierung

In Abschnitt 2.1.5 wurde eine Auswahl bestehender Vorgehensmodelle für Simulationsstudien vorgestellt sowie ein erster Blick auf die Stärken und Schwächen der Modelle für die automatische Modellgenerierung, -initialisierung, -adaption und -validierung geworfen. Grundlegend ist, nach Meinung des Autors, auch im Kontext der Automatisierung von Simulation die Nutzung von Vorgehensmodellen nötig. Dabei sind aber die speziellen Anforderungen der automatischen Modellgenerierung, -initialisierung, -adaption und -validierung adäquat zu beachten. Im Folgenden soll ein für diesen Anwendungsfall angepasstes Vorgehensmodell grob skizziert werden, das unter anderem folgende spezielle Aspekte fokussiert:

- Abkehr vom Projektcharakter, d.h. kontinuierliche Wieder- oder Weiterverwendung einzelner Modelle oder Teilmodelle, sowohl innerhalb als auch über Phasengrenzen des Produkt-/Produktionslebenszyklus hinweg
- konsequente durchgängige Automatisierung der Modellgenerierung, -initialisierung, Verifikation & Validierung, und Auswertung der Simulation für den Simulationsendnutzer
- Sicherstellung erweiterter Anpassungsmöglichkeiten für Simulationsexperten
- Trennung zwischen Baustein-/Generatorentwicklung und der Modellerstellung sowie -nutzung
    - Beachtung verschiedener Rollen (Softwareentwickler, Simulationsendnutzer, Simulationsexperte)
    - Einbeziehung von nötigen Schnittstellen im Entwicklungsprozess
    - Management von (Funktions-)Erweiterungen und geänderten technischen Anforderungen (z.B. Plattform, Simulator ...)
    - Entkopplung des Experimentdesigns sowie der nötigen Produktionsläufe und deren Auswertung
    - Verbergen von Simulatordetails/Implementierungsdetails

- Nutzung eines Konzeptmodells als zentrale Komponente, Fokussierung auf CMSD als Abbildung des Konzeptmodells
- Orientierung an den Grundsätzen der ordnungsgemäßen Modellierung (GoM).

Im Rahmen dieser Arbeit wird dabei vor allem auf die Simulation an sich und deren Durchführung sowie auf die Nutzung des CMSD Standards als Abbildung (formaler) Konzeptmodelle fokussiert. Der durchaus nicht unwichtige Teilaspekt der eigentlichen Entwicklung der Softwarekomponenten bzw. das dafür benötigte Vorgehensmodell wird nicht im Detail diskutiert werden. Eine Analyse und Bewertung der Vielzahl von möglichen Vorgehensmodellen zur Softwareentwicklung und das Aufstellen von Handlungsempfehlungen würde den Umfang der Arbeit erheblich überschreiten. Die durch den Autor im Rahmen der Entwicklung der Softwarekomponenten gesammelten Erfahrungen geben aber den Hinweis, dass Agile Vorgehensmodelle[14]die Anforderungen an Erweiterbarkeit und Flexibilität auch bzgl. der technologischen Rahmenbedingungen, z.B. unterschiedliche Simulatoren, besser als "klassische" Ansätze unterstützen. So wurden für die Entwicklung des Prototypen sowohl Ansätze des Extreme Programming (XP) vgl. [BA2004] als auch des Feature Driven Developments (FDD) vgl. [PF2002] aufgegriffen. Ebenso wird auf detaillierte Betrachtung von Testmethoden im Rahmen der Softwareentwicklung verzichtet. Hierbei ist grundlegend festzuhalten, dass ein Testen der entwickelten Funktionen aller Komponenten des Frameworks essentieller Bestandteil des Gesamtvorgehens sein muss. Eine ausreichende Softwarequalität aller Softwarekomponenten (vgl. 3.4), wie der der Modellgeneratoren (vgl. 3.4.2) oder des Statistikmonitors (vgl. 3.4.4) gesichert durch entsprechende Tests, wird im Folgenden unterstellt, da wie bereits andiskutiert einige der positiven Effekte des Konzepts zur automatischen Simulationsmodellgenerierung, -adaption und -validierung nur unter der Voraussetzung ausreichend gut getesteter und fehlerfrei funktionierender Softwarekomponenten realisiert werden können, z.B. Verbesserung der Klarheit der Modelle. Im Folgenden wird im Vorgehensmodell die gesamte Generatoren- und Schnittstellenentwicklung als Black-Box betrachtet (vgl. Abbildung 23, links unten).

[14] Agile Softwareentwicklung/Vorgehensmodell ist ein Sammelbegriff für verschiedene SW-Entwicklungsmethoden, die auf geringen bürokratischen Aufwand, wenige Regeln und iteratives Vorgehen abzielen vgl. Agiles Manifest [Be2001]

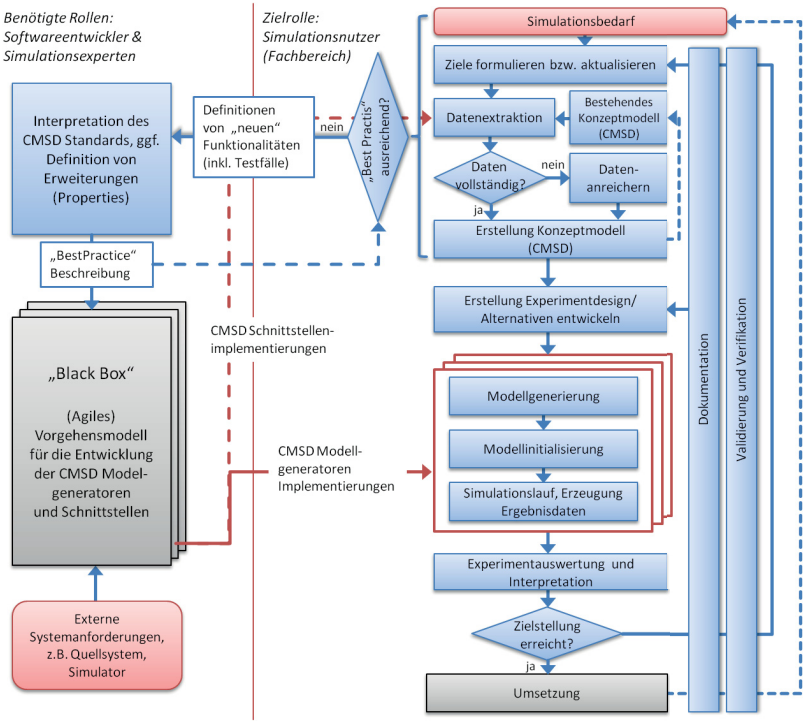


Abbildung 23: Angepasstes Gesamtvorgehensmodell für Simulationsstudien mit automatischer Modellgenerierung

Grundlage des Gesamtvorgehens und insbesondere auch der Generatorentwicklung bildet die in Form einer "Best Practice" beschriebene Interpretation des CMSD Standards, wie in Abschnitt 3.3 dokumentiert. Ergänzt wird diese Beschreibung durch Testfälle bzw. Beispielfälle in Form von CMSD Inputdaten und erwarteten Ergebnisdaten. Die dokumentierte CMSD Interpretation ergänzt durch die dokumentierten Beispieldaten stellt die zentrale Schnittstelle zwischen den Generatornutzern, d.h. dem Personenkreis, der konkrete Simulationsstudien durchführt, und den Simulationsexperten und Softwareentwicklern, die die Pflege der CMSD Interpretation und Implementierung des Modellgenerators für die gewünschten Simulationsumgebungen und der CMSD Schnittstellen in den betrieblichen Informationssystemen verantworten, dar. Die Dokumentation der CMSD Interpretation erfolgt in Anwendungsfällen (auch "User stories", "Use cases"), dies ermöglicht allen Beteiligten einen einfachen Einstieg in die komplexe Materie und stellt trotz des ggf. unterschiedlichen Backgrounds der Beteiligten ein einheitliches Verständnis sicher. Neue Anforderungen der Simulationsnutzer, die sich während der Systemanalyse bzw.

Beschreibung auf Grund spezieller Unternehmensanforderungen bzw. bisher nicht in der CMSD "Best Practice" betrachteten Eigenschaften des "realen" Systems ergeben, sollten als Anwendungsfälle beschrieben und in die CMSD Interpretation integriert werden. Bei der Integration ist dabei besonders auf Konsistenz zu der bestehenden Interpretation zu achten, des Weiteren ist auf eine gewisse Allgemeingültigkeit, der geringen Nutzung von zusätzlichen Properties und vor allem Verständlichkeit und Nachvollziehbarkeit zu achten. Diese essentielle Aufgabe benötigt zwingend die Einbeziehung von Simulationsexperten. Die so definierten neuen Funktionen/Anforderungen können im Zuge der Softwareentwicklung je nach konkret gewähltem Vorgehensmodell z.B. als "User Story" im Falle von XP oder als "Feature" im Fall von FDD aufgefasst werden. Weitere Trigger für die Softwareentwicklung können "externe Systemanforderungen" darstellen. Diese können sehr vielfältig sein. So sind hier Anforderungen subsumiert, die sich überwiegend aus der im konkreten Fall vorhandenen IT Infrastruktur ergeben, z.B. dem eingesetzten ERP oder MES, das als Datenquelle dienen soll oder der präferierten Simulationsumgebung.

Die eigentliche Nutzung der Simulation umfasst die typischen Phasen (Aufgabenanalyse, Modellformulierung, Modellimplementierung, Modellüberprüfung, Modellanwendung [RSW2008, S. 29]), die bereits in Abschnitt 2.1.5 diskutiert wurden. Dabei werden aus den etablierten Vorgehensmodellen die für die automatischen Simulationsmodellgenerierung, -adaption und -validierung geeignetsten Konzepte übernommen. So wurde aus dem Vorgehensmodell nach Sargent [Sa1982] die strikte Trennung von Konzeptmodell und dem implementierten Simulationsmodell sowie die Betonung der Bedeutung der Validierung und Verifikation in allen Phasen und damit einhergehend der Möglichkeit von Iterationen übernommen. Neben der bereits in Sargents Vorgehensmodell erkennbaren Trennung von Konzept- und Simulationsmodell wird dem Vorgehensmodell von Law [La2007] die Entkopplung des Experimentdesign, der Ergebnisauswertung und -interpretation sowie der Modellinitialisierung entnommen. Des Weiteren betont Law die Bedeutung der Dokumentation in der Umsetzungsphase aber auch im Allgemeinen. Schließlich wird aus dem VDI Vorgehensmodell neben den auch teilweise hier abgebildeten, bereits geschilderten Aspekten, vor allem die Überprüfung der Zielerreichung und der dadurch bedingten Iterationen sowie die explizite Betrachtung von Alternativen aufgenommen.

Auch das hier vorgestellte angepasste Vorgehensmodell kann eine gelebte Kultur zur Einbindung der Simulation in weite Phasen der Planung und Steuerung der Produktion nicht ersetzen, sondern nur positiv unterstützen. Begleitende organisatorische Maßnahmen werden im Folgenden zwar nicht im Detail betrachtet, sind aber nach Meinung des Autors für eine optimale Nutzung von simulationsbasierten Methoden generell unabdingbar.

Allgemein ist das dargestellte Vorgehensmodell nur als grober Rahmen zu sehen, die Phasen oder Gruppen von Phasen können jederzeit mehrfach durchlaufen werden, bis z.B. valide Ergebnisse vorliegen bzw. bis alle gewünschten Alternativszenarios abgedeckt sind. Allgemein gilt, dass die Simulationsstudie von allen Beteiligten nicht als einmaliges Projekt aufgefasst werden sollte. Es gilt ferner, alle Phasenergebnisse so zu dokumentieren und archivieren, dass sie bei erneutem Simulationsbedarf jederzeit verfügbar sind. Das so gewonnene Datenmaterial/Wissen kann neben dem reinen Dokumentationszweck zur Senkung von Aufwänden (Zeit und Kosten) beitragen und die Qualität aller (Phasen-) Ergebnisse verbessern.

Eine konkrete Simulationsstudie kann durch verschiedenste betriebliche Ereignisse nötig werden. So kann sowohl das Erreichen eines Meilensteins in der Planung einer neuen Anlage, das Erkennen von Engpässen in der aktuellen Produktion, ein Maschinenausfall oder schlicht das Aufstellen eines neuen Produktionsplans einen Simulationsbedarf auslösen. Unabhängig vom auslösenden Ereignis und den damit verbundenen Rahmenbedingungen unterscheidet sich das grundlegende Vorgehen für die automatische Modellgenerierung, -initialisierung, -adaption und -validierung kaum, auch wenn einzelne Phasen durchaus deutlich unterschiedlich aufwändig sein können. So ist beispielsweise im Fall eines neuen Produktionsprogramms anzunehmen, dass weite Teile des Konzeptmodells (vgl. Abschnitt 2.1.5) bis auf aktuelle Lastdaten bereits vorliegen und auch erfolgreich validiert wurden. Somit sind ggf. nur diese Lastdaten aus den entsprechenden Quellsystemen (z.B. ERP) zu extrahieren und in das bestehende CMSD zu integrieren, d.h. es erfolgt eine Adaption des Konzeptmodells, die weiteren Phasen des Vorgehensmodells sind davon vollständig unabhängig.

Die erste Phase im Vorgehensmodell, nach der Definition der Ziele, dient der Erstellung des CMSD basierten (formalen) Konzeptmodells. Zunächst werden dazu mittels geeigneter Schnittstellen die Daten aus den entsprechenden Quellsystemen extrahiert (vgl. Abbildung 8) und/oder auf bestehende CMSD Daten zurückgegriffen. Die Implementierung der Schnittstellen zur Datenextraktion stellt zwar je Quellsystem einen einmalig hohen Aufwand dar, im Rahmen der angestrebten integrierten Nutzung der Simulation in weiten Bereichen des unternehmerischen Handelns kann aber durchaus ein positives Nutzenverhältnis unterstellt werden. Fehler durch fehlerhafte Datenübernahme, z.B. Tippfehler bei der manuellen Datenerhebung, werden vermieden. Die so ermittelten Basisdaten durchlaufen in jedem Fall eine automatisierte Vollständigkeitsprüfung, d.h. die Daten werden daraufhin geprüft, ob alle für eine Simulationsmodellgenerierung nötigen Informationen vorhanden sind. Eine inhaltliche fachliche Prüfung findet hierbei ausdrücklich nicht statt. Die inhaltliche fachliche Prüfung der Daten (Datenvalidierung vgl. Abschnitt 2.1.5 Vorgehensmodell von Sargent [Sa1982]) erfolgt parallel im Rahmen der permanenten Validierung und Verifikation.

Dabei können sowohl manuelle Prüfungen durch den Simulationsnutzer als auch automatische Prüfungen Anwendung finden. In diesem Kontext sei vor allem auf die rechnerunterstützten Plausibilisierungsmethoden mittels Filterklassen nach Müller-Sommer [MS2013] verwiesen.

Sowohl die Vollständigkeitsprüfung als auch die Datenvalidierung können einen Bedarf an zusätzlichen Informationen aufzeigen, dieser kann durch zusätzliche Datenextraktionen, manuelle oder automatische Datenanreicherung gedeckt werden. Die manuelle Datenanreicherung des Konzeptmodells muss durch geeignete Softwarelösungen möglichst aufwandsoptimal gestaltet werden, dies kann durch geeignete Nutzeroberflächen (z.B. ein Webfronend) unterstützt werden (vgl. Abschnitt 3.4.3). Eine (teil-) automatische Datenanreicherung kann auf verschiedenste Weise realisiert werden, so bieten sich vor allem regelbasierte Lösungen, auf Basis von bspw. Expertenwissen oder Daten anderer Simulationsläufe an. In dem im Rahmen dieser Arbeit entwickelten Framework wird allein die Komponente zur manuellen Datenanreicherung implementiert, konkrete Vorschläge für Methoden zur automatischen Datenanreicherung sind nicht Bestandteil dieser Arbeit. Gleichwohl kann das von Müller-Sommer in seiner Dissertation [MS2013] betrachtete Simulationsgerüst als ein erster Schritt zur regelbasierten Datenanreicherung aufgefasst werden, bedarf aber weiterführender wissenschaftlicher Betrachtung.

Die im Rahmen der ersten Phase entstandenen CMSD Daten, die das aktuelle Konzeptmodell darstellen, können direkt archiviert werden und stehen somit bei erneutem Simulationsbedarf jederzeit zur Verfügung.

In der Phase "Erstellung Experimentdesign/ Alternativen entwickeln" müssen die Experimentparameter, wie beispielsweise der Simulationszeitraum oder die Anzahl benötigter Replikationen festgelegt werden. Des Weiteren ist aufbauend auf dem erzeugten CMSD basierten (Basis-) Konzeptmodell die Definition von Alternativen möglich, dazu werden die CMSD-Konzeptmodelldaten gezielt manipuliert, indem einzelne Teile verändert, gelöscht oder ergänzt werden, die technische Realisierung ist analog der Datenanreicherung zu sehen, so ist wiederum sowohl eine manuelle als auch automatische Alternativengenerierung [SBS2012] vorstellbar. Unabhängig der Erzeugungsmethode sind die Ergebnisse dieser Phase die um Experimentdaten angereicherten, simulatorneutralen CMSD basierten Konzeptmodelle. Hierbei wird jede Alternative in Form einer CMSD XML Datei gespeichert.

Die CMSD XML Dateien sind alleinige Datenbasis der eigentlichen Simulation, d.h. den Subphasen Modellgenerierung, Modellinitialisierung, Simulationslauf und Ergebnisdatenerzeugung. Die technische Realisierung wird in Abschnitt 3.4.2 detailliert beschrieben. Eine Basisanforderung an das Vorgehen ist die Simulatorneutralität, aus

diesem Grund werden analog zu den Konzeptmodellen alle Ergebnisdaten eines Simulationslaufes in CMSD als neutrales unabhängiges Format gespeichert (vgl. Abschnitt 3.3.9 und 3.4.4). Diese neutralen Daten können je Lauf, für einzelne Alternativen oder im Vergleich mehrerer Alternativen, mittels der Simulationsoutputanalyse (vgl. Abschnitt 2.1.7) bewertet werden. Je nach Performance der Ergebnisse erfolgen weitere Planungen und somit ggf. Simulationen oder eine entsprechende Umsetzung.

Wie bereits angemerkt, ist die Datenvalidität, die für die zur automatischen Simulationsmodellgenerierung, -adaption und -validierung wichtigste, wenn auch nicht einzige Teilaufgabe im Zuge der Validierung und Verifikation. Sargent unterscheidet in seinem Vorgehensmodell neben der Datenvalidität drei weitere Fälle der Validierung bzw. Verifikation: die Konzeptmodellvalidierung, die Verifikation des (Computer-) Simulationsmodells und die Validierung anhand der Simulationsergebnisse. Die Konzeptmodellvalidierung ebenso wie die Verifikation weist im Vergleich zu nicht automatisierten Simulationsstudien einen deutlich abweichenden Charakter auf, beide können unter der Annahme der Richtigkeit aller Komponenten des Frameworks vorab direkt unterstellt werden. So ist bei gegebener Datenvalidität und unter Einhaltung der Best Practice Konzeptvalidität gegeben. Die (Computer-) Simulationsverifikation kann entfallen, da unter der vorab getroffenen Annahme der gesicherten Qualität der Modellgeneratoren (Teil des Softwareentwicklungsprozesses) aus einen Konzeptmodell immer ein entsprechendes formal korrektes Simulationsmodell generiert wird.

Da die Datenvalidierung in der Praxis durchaus ihre Grenzen hat, ist die Validierung der Ergebnisse umso wichtiger einzustufen [RSW2008, S.131]. Zu beachten ist, dass möglicherweise entdeckte Abweichungen sowohl auf fehlende oder falsche Daten hinweisen können, als auch auf Fehler in der Anwendung der Best Practice oder Fehler der Generatorsoftware. Da im Vorgehensmodell der Simulationsnutzer nicht zwingend ein Simulationsexperte sein muss und da eine möglichst umfassende Automatisierung angestrebt wird, sind nicht alle üblicherweise in der Praxis anzutreffenden Validierungsverfahren (siehe Tabelle 8) gleich gut geeignet. So sind besonders auf Ergebnisdaten basierende Verfahren zu präferieren und hierbei insbesondere Verfahren die selbst eine gewisse Automatisierbarkeit aufweisen, wie z.B. der Vergleich mit aufgezeichneten Daten [RSW2008, S. 111] oder die Validierung von Vorhersagen [RSW2008, S.109f]. Eher ungeeignet erscheinen Verfahren, die direkten Simulatorzugriff bzw. detailliertes Implementierungswissen benötigen wie beispielsweise der Test der internen Validität [RSW2008, S.105f] oder das strukturierte Durchgehen [RSW2008, S.104f]. Die konkrete Wahl der Verfahren ist aber stark vom Anwendungsfall, Erfahrung der Beteiligten usw. abhängig.

Tabelle 8: Praxisrelevante V&V Techniken (in Anlehnung an [RSW2008, S96]); sowie deren Eignung zur Automatisierung (Ampelskala)

Bezeichnung	Bezeichnung	Bezeichnung	Bezeichnung
Animation	Grenzwerttest	Strukturiertes Durchgehen	Ursache-Wirkungs-Graph
Begutachtung	Monitoring	Test der internen Validität	Validierung im Dialog
Dimensionstest	Schreibtischtest	Test von Teilmodellen	Validierung von Vorhersagen
Ereignisvaliditätstest	Sensitivitätsanalyse	Trace-Analyse	Vergleich mit anderen Modellen
Festwerttest	Statistische Techniken	Turing-Test	Vergleich mit aufgezeichneten Daten

Originär neue Validierungstechniken wurden im Rahmen dieser Arbeit nicht betrachtet, Ausführungen zu Möglichkeiten der Automatisierung und Einbindung in das Framework werden in Abschnitt 3.4 angerissen. Da im vorliegenden Vorgehensmodell alle Ergebnisse ausdrücklich zur Wieder- bzw. Weiterverwendung entworfen werden, wird empfohlen die V&V Aktivitäten auch im Rahmen der Umsetzung konsequent weiterzuführen. Konkret kann gesagt werden, dass eine Kontrolle der Ergebnisse mit der realen Umsetzung (ggf. auch periodisch) helfen kann, die Qualität der zukünftige Ergebnisse zu erhöhen. Dabei hilft gerade die Automatisierung der Techniken den Aufwand einzudämmen.

Automatische Modellgenerierung im Allgemeinen und die Nutzung des hier vorgestellten Vorgehensmodells im Speziellen haben positiven Einfluss auf die Grundsätze der ordnungsgemäßen Modellierung (vgl. Abschnitt 2.1.4).

So werden potentielle Fehler, beispielsweise bei der manuellen Datenübernahme, Implementierung der Modelle im Simulator oder Kennzahlenberechnung usw. durch Automatisierung vermieden, wodurch die Richtigkeit des Konzept- und Simulationsmodells verbessert wird. Des Weiteren kann eine höhere Relevanz der Modelle vor allem durch die Anwendung der Best Practice Beschreibung und entsprechender Nutzerschnittstellen während der Konzeptmodellerstellung erreicht werden. Ebenso ist zu erwarten, dass trotz der einmaligen Aufwände zur Entwicklung der Generatoren, Schnittstellen usw. die Wirtschaftlichkeit steigt. Dies ist nicht allein auf die Zeit und Kosteneinsparungen bei der automatischen Modellgenerierung direkt, sondern ebenso auf die verbundene Automatisierung, Qualitätsverbesserung und Standardisierung weiterer Prozesse wie der Datenextraktion, der Alternativentwicklung,

Ergebnisauswertung usw. zurückzuführen. Die Klarheit, Vergleichbarkeit und der systematische Aufbau der Konzeptmodelle aber insbesondere auch der Simulationsmodelle wird durch die durchgehende Standardisierung und Automatisierung verbessert.

Eine zunehmende Verbreitung des Ansatzes, d.h. bei Erreichen einer kritischen Nutzerzahl und Anwendungshäufigkeit, können durch stetige Verbesserung und Erweiterung der Best Practice Lösung und aller Frameworkkomponenten, inklusive der Schnittstellenimplementierungen für verbreitete betriebliche IT Systeme, tendenziell erhöhte Nutzeneffekte realisiert werden.

Wie den Ausführungen zum hier vorgeschlagenen Vorgehensmodell zu entnehmen, ist das Konzeptmodell in Form von CMSD essentielle Grundlage des Gesamtkonzepts. Bevor im Abschnitt 3.3 die konkrete Interpretation des CMSD Standards ausführlich diskutiert wird und anschließend Konzepte der (Software-) Komponenten und entsprechende Prototypen des Frameworks vorgestellt werden, wird zunächst die als Lücke vieler Ansätze identifizierte Modellierung dynamischen Verhaltens thematisiert.

3.2 Modellierung dynamischen Verhaltens - Steuerstrategien

In allen realen Fertigungssystemen müssen während des Betriebs verschiedenste Entscheidungen im Rahmen der Produktionsplanung und Steuerung (PPS) getroffen werden. Für die Simulation von besonderem Interesse sind operative Entscheidungen, d.h. im Rahmen der Simulation steht der Aspekt der Steuerung (auch Produktionsprozessplanung [15]) im Fokus der Betrachtungen. Die planerischen Entscheidungen, z.B. bzgl. des Produktionsprogramms, "Make or Buy" usw. spiegeln sich bereits in den Eingangsdaten, im Speziellen den Lastdaten und Organisationsdaten (vgl. Abschnitt 2.1.6), der Simulation wieder.

Gerade aber die Modellierung und Generierung der Steuerung stellt eine Lücke innerhalb der meisten Modellgenerierungsansätze dar [Se2005, S.30; BRA2008, S. 441]. Die Steuerung ist aber einer der wichtigsten Faktoren für erfolgreiche Simulationsprojekte, so ist nach Law und McComas eines der grundlegenden Elemente für den Erfolg eines jeden Simulationsprojektes in ausreichend guten Informationen über Abläufe und Steuerungslogiken zu sehen [LC1991, S.21].

[15] Produktionsprozessplanung lässt sich als "Systematisch vorbereitete Festlegung der (kalender-) zeitlichen und örtlichen Reihenfolge von Aktionen (Be- und Verarbeitungs- sowie hiermit verbundene Transport- und Lagervorgänge) zur Durchführung von Produktionsaufträgen für Vor- und Endprodukte bei grundsätzlich gegebenem Potenzialbestand unter Beachtung des Wirtschaftlichkeitsprinzips und von Anforderungen aus dem Humanbereich." definieren [Ga2013].

Nach Selke [Se2005] lassen sich sämtliche Steuerungsentscheidungen, die für die Simulation direkt von Interesse werden, in eines von 4 Strategieclustern (vgl. Abbildung 24) einordnen. Selke stellt aber bereits fest, dass für die Auftragsfreigabe, die allein den Bearbeitungsbeginn des Auftrags definiert, "Häufig [...] übergeordnete Systeme zur Bestimmung dieses Zeitpunktes eingesetzt" [Se2005, S48] werden. Dieser Aussage folgend wird die Auftragsfreigabe, die ohne Zweifel erheblichen Einfluss auf die Performance der Fertigung hat, als simulationsexterne Funktion betrachtet, der Freigabezeitpunkt wird im Folgenden konsequent als ein simpler Teil der Eingangsdaten der Simulation interpretiert.

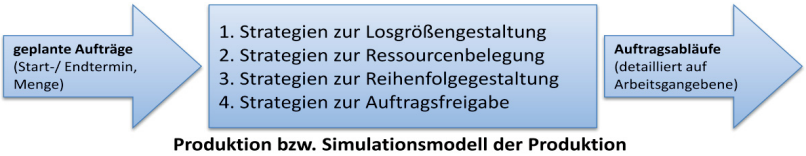


Abbildung 24: Relevante Cluster von Strategien und Abläufen der Produktionssteuerung [Se2005, S.45]

Prinzipiell beinhaltet jedes Cluster ein breites Spektrum einzelner Strategien (Abläufe), wobei durch Kombination oder Variationen weitere Strategien gebildet werden können, so ist zu bemerken, dass "in der Praxis eine unendliche Anzahl an Strategien bzw. Abläufen entstehen kann" [Se2005, S. 45].

3.2.1 Dezentrale Steuerung mittels Prioritätsregeln

Prinzipiell kann zwischen zentraler oder dezentraler Steuerung einer Fertigung unterschieden werden, wobei für die Simulation im engeren Sinne die dezentralen Steuerungsmethoden interessanter sind. Dies basiert auf der einfachen Annahme, dass sich zentrale Steuerungsentscheidungen in den Eingangsdaten der Simulation, z.B. anhand von Freigabeterminen oder Prioritäten wiederspiegeln, dezentrale Methoden aber direkt als Teil des Modells aufzufassen sind. Die Steuerungslogiken werden normalerweise im Simulationsmodell algorithmisch bzw. als Attribut vordefinierter Bausteine abgebildet. Im Rahmen der Automatisierung der Modellerstellung ist diese Vorgehensweise nicht ausreichend, somit müssen Methoden geschaffen werden, die es erlauben Regeln abzubilden sowie diese auf geeignete Weise im automatischen Modellgenerierungsprozess zu integrieren. Ein Begriff, der oft synonym mit dezentralen Steuerungsmethoden genutzt wird, ist die Entscheidungs- oder Prioritätsregel [BS2011, S.95; Pi2002].

Prioritätsregeln bilden anhand von Merkmalen von Aufträgen Prioritäten bzw. Prioritätenlisten. Auf deren Basis können Entscheidungen, z.B. über Reihenfolgen, Routing oder Losgrößensplitting getroffen werden (siehe Abbildung 24). Prioritätsregeln

sind prinzipiell für alle Strategiecluster gleichermaßen anwendbar, zur Vereinfachung wird im Folgenden, wenn nicht explizit anders angegeben, von Reihenfolgeentscheidungen vor Bearbeitungsstationen ausgegangen.

Nebl definiert die Prioritätsregeln für Reihenfolgeentscheidungen als „[…] Vorschriften, nach denen festgelegt wird, in welcher Reihenfolge die vor einer Maschine in einer Warteschlange befindlichen Aufträge bearbeitet werden.“ [Ne2007, S. 691]. Eine Auswahl typischer Regeln kann in [Ne2007, S. 691 f.] oder [FFS2011, S. 760 f.] gefunden werden. Drei Beispielregeln und deren Auswirkungen auf die Reihenfolge von Aufträgen sind Abbildung 25 zu entnehmen.

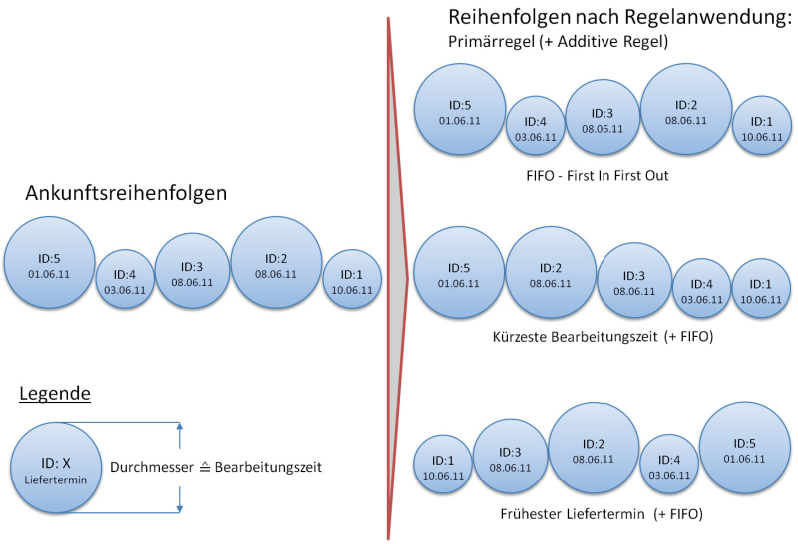


Abbildung 25: Beispielwirkung von Prioritätsregeln (in Anlehnung an [BS2011, S. 95]

Prioritätsregeln lassen sich anhand der herangezogenen Daten klassifizieren, so kann zunächst zwischen Regeln, die allein lokale Daten (z.B. Füllstand des aktuellen Puffer, Bearbeitungszeit auf der aktuellen Station) und Regeln, die globale Daten (z.B. Füllstände aller Puffer des Systems) nutzen, unterschieden werden. Eine weitere Klassifizierung ist beispielsweise anhand der Charakteristik der ausgewerteten Merkmale möglich [BS2011, S.95f], vgl. Tabelle 9.

Tabelle 9: Klassen von Prioritätsregeln (in Anlehnung an [BS2011, S.95])

Merkmalscharakteristik	Beispielregel
Eintrittszeitorientiert	FIFO - First In First Out (unveränderte Reihenfolge) [im Fertigungssystem = global; in der aktuellen Station = lokal]
Bearbeitungszeitorientiert	SPT - Shortest Processing Time (kürzeste Operationszeit)
Rüstzeitorientiert	SST - Shortest Setup Time (Rüstoptimal)
Terminorientiert	EDD - Earliest Due Date (geringster Liefertermin) ERD - Earliest Release Date (frühester Freigabetermin)
Kundenprioritätsorientiert	Höchste Kundenpriorität
Wert- / Kostenorientiert	Maximaler Deckungsbeitrag Maximale bisher erzeugte Kosten
Kombinationen	WINQ (niedrigster Arbeitsvorrat an der nächsten Maschine) SL - Slack Time; COVERT - Cost over Time

Prioritätsregeln können zusätzlich auf verschiedene Arten kombiniert werden, so sind Zusammensetzungen ebenso wie bedingte oder Multi-Level Prioritätsregeln möglich [Mö2006, S.21f].

Bei den zusammengesetzten Prioritätsregeln werden mehrere Prioritätsregeln entweder additiv, multiplikativ oder alternativ zu einer Einzigen zusammengefasst, d.h. einzelne Prioritäten werden zu einer verrechnet. In der so entstandenen zusammengesetzten Prioritätsregel sollen die positiven Eigenschaften aller genutzten Subprioritätsregeln miteinander verbunden sowie einzelne negative Eigenschaften abgeschwächt werden [Ml2006, S. 22], [FFS2011, S. 761].

Bei der Nutzung bedingter Prioritätsregeln wird abhängig von aktuellen Fertigungssystemparametern, z.B. der Auslastung des Fertigungssystems, eine entsprechende Prioritätsregel gewählt [Mö2006, S. 22].

Schließlich werden bei so genannten Multi-Level-Prioritätsregeln ggf. verschiedene Prioritätsregeln sequentiell angewendet. So wird im Falle einer unklaren Entscheidung der ersten Regel in einem Folgeschritt eine zweite Regel auf die Menge der "Gewinner" der ersten Regel angewendet. So kann beispielsweise im Fall der Anwendung der "kürzesten Operationszeit" Regel der Fall auftreten, dass zwei oder mehrere zu priorisierende Aufträge die kürzeste Operationszeit aufweisen, in diesem Fall kann für diese Aufträge keine eindeutige Bearbeitungsreihenfolge ermittelt werden. Bei Multi-Level-Prioritätsregeln wird in solch einem Fall in einem zweiten Schritt eine weitere Regel, z.B. FIFO auf alle Aufträge mit der kürzesten Operationszeit angewendet und nach dieser die Reihenfolge abschließend definiert [Mö2006, S. 22]. Theoretisch können

hierbei beliebig viele Regeln aufeinander folgen, die Auswertung endet, wenn es einen eindeutigen priorisierten Auftrag gibt oder alle Regeln angewendet wurden. Da bis auf die (technische) Ankunftsreihenfolge (FIFO) die meisten Regeln nicht zwingend einen eineindeutigen Auftrag priorisieren, sind Multi-Level-Prioritätsregeln z.B. mit FIFO als letzter Instanz durchaus sinnvoll [BSS2013, S.3-6].

3.2.2 Ansätze zur Abbildung von Steuerstrategien in der Modellgenerierung

Für die automatische Modellgenerierung problematisch ist, dass Informationen über Strategien im Allgemeinen und im Speziellen über Prioritätsregeln üblicherweise nicht in den betrieblichen IT-Systemen, z.B. der Systeme der Digitalen Fabrik oder ERP-Systeme usw. hinterlegt, oder gar nur implizit in den Köpfen der Werker vorhanden sind [BS2010b, S.547; RG2008, S.302f]. Somit ist für die grundlegende Modellerstellung meist zusätzlich eine Ermittlung der Strategien aus bestehenden Datenbeständen, vor allem aus Daten der Betriebsdatenerfassung (BDE) oder eine gesonderte Abfrage der Regeln oder gar Modellierung der Steuerungslogik nötig.

In diesem Kontext lassen sich 3 Klassen von Ansätzen unterscheiden:

1. das Erkennen und Nutzen von kombinierbaren Basisregeln (microfunctions [Wi1999]),
2. die Modellierung der Logik in einer systemneutraleren Sprache und Codegenerierung sowie
3. die Approximation der Regeln.

Vor allem der erste Ansatz, in dem auch die im Folgenden betrachteten Arbeiten von Selke [Se2005] sowie Reinhart und Gyger [RG2008] einzuordnen sind, basiert auf der generellen Grundannahme, dass sich alle benötigten Steuerungslogiken/Strategien aus der Kombination einer endlichen Zahl von einfachen Bausteinen, so genannten "microfunctions" ggf. unter Angabe von Parametern (z.B. Gewichten) erzeugen lassen. Erstmals wurden sehr fein granulare "microfunctions" zur Abbildung von komplexem Verhalten im Kontext der Simulation im Rahmen eines datenbankbasierten Modellierungsframeworks von Wiedemann genutzt [Wi1999], auch Schönherr und Rose nutzen ähnliche Ansätze im Kontext der Modellgenerierung aus SysML Diagrammen [SR2011, S. 2210-2213]. Wiedemanns sehr fein granulare "microfunctions" sind im Kontext der datengetriebenen automatischen Modellgenerierung weniger gut geeignet [Se2005, S.29], somit haben sich weitestgehend die vorab vorgestellten Prioritätsregeln als kleinste Bausteine etabliert. Diese Bausteine müssen entsprechend implementiert sein, d.h. während der Modellgenerierung werden nur im Simulator bestehende Regeln ausgewählt und ggf. kombiniert bzw. parametriert. Eine Auswahl an Regeln und Kombinationsmöglichkeiten für diese ist im Modellgenerierungsframework umgesetzt (vgl. Abschnitt 3.3).

Die so verfügbaren Regeln lassen sich auch ohne automatische Erkennung nutzen, indem der Simulationsnutzer die Regel angibt. Somit stellt dieser Teil des Ansatz eine sehr einfache neutrale textuelle Modellierungssprache dar (im einfachsten Fall die Angabe von Regelnamen in einem entsprechenden Frontend) (vgl. Abschnitt 3.4.3).

Für das Erkennen der Strategien aus BDE-Daten müssen zunächst die Orte, an denen Steuerungsentscheidungen getroffen werden, erkannt werden. Im Fall von Reihenfolgeentscheidungen sind dies üblicherweise Puffer vor einer kleinen Zahl von Stationen (z.B. Engpassstationen). Die automatische Erkennung gestaltet sich relativ einfach. Je nach Strategiecluster wird nach der Wirkung von Steuerungsregeln gesucht, z.B. sich überholende Aufträge oder Losgrößenänderungen [Se2005, S.68-75].

Im zweiten Schritt kommen Verfahren der Musterkennung zum Einsatz, Selkes Ansatz [Se2005] erkennt dabei nur direkt vorab definierte Strategien/Regeln, Reinhart und Gyger [RG2008] erweitern diesen Ansatz so, dass auch Multi-Level-Prioritätsregeln erkannt werden können. Für weitere Details sei auf die entsprechenden angegebenen Veröffentlichungen verwiesen. Die gesamte Erkennung der Regeln kann aber im Zuge der Modellgenerierung als externer Prozess aufgefasst werden, der zusätzliche Eingangsdaten, die Regel oder Kombinationen liefert.

Der zweite Ansatz der neutralen Beschreibung und Codegenerierung wird bisher kaum aufgegriffen und soll an dieser Stelle auch nicht weiter thematisiert werden.

Der dritte Ansatz zur Abbildung der Steuerungsregeln ist die Approximation der Regeln. Auch dieser Ansatz basiert auf der Verarbeitung von BDE-Daten, erkennt aber keine Regeln sondern will allein deren Verhalten nachbilden [BS2011, S.94; BSS2013, S.2].

Der im Folgenden im Detail beschriebene Ansatz zur Approximation unter Nutzung von künstlichen Neuronalen Netzen (KNN) wurde in [BS2011] und [BSS2013] veröffentlicht, eine erweiterte Evaluierung des Ansatzes erfolgte in [Ba2011].

3.2.2.1 Prioritätsregeln als Funktionen

Erste Grundannahme des Ansatzes ist, dass jede dezentrale Steuerstrategie, vor allem die bereits geschilderten Prioritätsregeln und deren Kombinationen, als Funktion aufgefasst werden kann, wobei die Argumente der Funktion Merkmale der Aufträge und das Funktionsergebnis die Priorität des Auftrags darstellt. Zur Ermittlung des nächsten Auftrags an einer Station müssen folglich für alle wartenden Aufträge die Funktionswerte ermittelt werden, der Auftrag mit dem höchsten Wert wird anschließend bearbeitet.

Zur Veranschaulichung sind im Folgenden die in Abbildung 25 aufgeführten einfachen Prioritätsregeln FIFO (Formel 2), kürzeste Bearbeitungszeit (Formel 3) und frühester Liefertermin (Formel 4) vereinfacht als Formel der Gestalt (vgl. Formel 1):

Formel 1: Prioritätsregeln → Prioritätsfunktion

$$P_i = \sum(g_j * a_{ji})$$

Legende: P_i: Priorität des Auftrags i
aji: Attribute j des Auftrags i
gj: Gewicht des Attribute j

dargestellt. Zur Wahrung der Übersichtlichkeit wird hierbei auf die für diese Regeln relevanten Auftragsmerkmale: Ankunftszeit (t_{arr}), Liefertermin (t_{dd}) und Bearbeitungszeit (t_{pt}) eingeschränkt.

Formel 2: Darstellung der FIFO Regel als Funktion

$$P_i = \frac{1}{t_{arr\,i}} = \left(1 * \frac{1}{t_{arr\,i}}\right) + (0 * t_{pt\,i}) + \left(0 * \frac{1}{t_{dd\,i}}\right)$$

Formel 3: Darstellung der kürzesten Bearbeitungszeit Regel als Funktion

$$P_i = \frac{1}{t_{pt\,i}} = \left(0 * \frac{1}{t_{arr\,i}}\right) + \left(1 * \frac{1}{t_{pt\,i}}\right) + \left(0 * \frac{1}{t_{dd\,i}}\right)$$

Formel 4: Darstellung der frühester Liefertermin Regel als Funktion

$$P_i = \frac{1}{t_{dd\,i}} = \left(0 * \frac{1}{t_{arr\,i}}\right) + (0 * t_{pt\,i}) + \left(1 * \frac{1}{t_{dd\,i}}\right)$$

Des Weiteren sind Verknüpfungen von Regeln ebenso als Funktion abbildbar, beispielsweise eine gewichtete additive Verknüpfung der kürzesten Bearbeitungszeit und der frühesten Liefertermin Regel (vgl. Formel 5)

Formel 5: Darstellung der Verknüpfung der kürzesten Bearbeitungszeit und frühester Liefertermin Regel als Funktion

$$P_i = \left(0 * \frac{1}{t_{arr\,i}}\right) + \left(0.4 * \frac{1}{t_{pt\,i}}\right) + \left(0.6 * \frac{1}{t_{dd\,i}}\right)$$

3.2.2.2 Künstliche Neuronale Netze

Cybenko [Cy1989] bewies bereits 1989, dass bestimmte künstliche neuronale Netze, im Speziellen Multi Layer Perceptrons mit sigmoiden Aktivierungsfunktionen universelle Funktionsapproximatoren darstellen (Cybenko-Theorem).

Nach Zell werden als künstliche neuronale Netze (KNN) informationsverarbeitende Systeme bezeichnet, die aus einer Menge einfacher Einheiten, wie z.B. Zellen, Neuronen oder auch Units, bestehen. Diese senden sich Nachrichten über gerichtete Verbindungen durch Aktivierung von Zellen zu [Ze1997, S.23]. Für den durch das menschliche Hirn inspirierten Ansatz der KNN sind in der Literatur verschiedenste

unterschiedliche Lernverfahren, Architekturen usw. dokumentiert. Diese weisen jeweils unterschiedliche Eigenschaften, Anwendungsgebiete, Stärken und Schwächen auf. Allen neuronalen Netzen gemein sind die 3 Hauptbestandteile: Neuronen, Verbindungen sowie Lernverfahren [Ze1997, S.72ff.], [Kr2013, S.37ff]. Für weiterführende Informationen sei an dieser Stelle auf die entsprechende Literatur verwiesen, z.B. Zell "Simulation Neuronaler Netze" [Ze1997] oder Kriesel "Ein kleiner Überblick über Neuronale Netze" [Kr2013].

Einer der klassischen Vertreter der KNN ist das bereits angesprochene Multi Layer Perceptron (MLP), welches sich aus mehreren Single Layer Perceptrons (SLP) aufbaut. Ein SLP ist ein Feedforward Netz, d.h. alle Verbindungen verlaufen von einer niederen zu einer höheren Schicht [RG1994, S. 48]. Es besteht nur aus einer trainierbaren Schicht, die der Ausgabeschicht entspricht, sowie der Eingabe-(Input) Schicht. Ein MLP besitzt N trainierbare Schichten sowie eine Eingabeschicht, wobei jede Schicht als Input ihrer Nachfolgeschicht dient, trainierbare Schichten, die nicht der Ausgabe dienen werden als verdeckte Schichten (Hidden Layer) bezeichnet (vgl. Abbildung 26). MLPs sind prinzipiell vollvermascht, d.h. jedes Neuron einer niederen Schicht besitzt eine Verbindung zu jedem Neuron der nächsten Schicht (Shortcuts über Schichten hinweg sind ausgeschlossen), die Eingaben und Aktivierungen der Neuronen können zudem nur Werte zwischen -1 und 1 (bzw. 0 und 1) annehmen, gebräuchlich sind sigmoide Aktivierungsfunktionen [Kr2013, S.76f; Ze1997, S.97f]. Als Lernverfahren kommen Verfahren des überwachten Lernens, bei denen zumindest eine Menge von Eingabemustern sowie deren korrekte Ergebnisse verfügbar sind (Trainingsdaten), zum Einsatz [Kr2013, S.55].

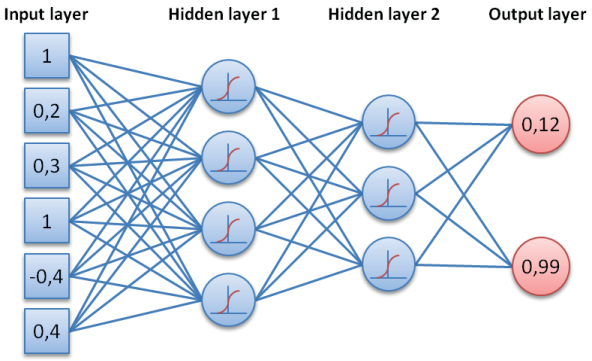


Abbildung 26: Beispiel eines MLP mit 2 verdeckten Schichten (in Anlehnung an [BS2S013, S.5]

Ein typisches und im Folgenden genutztes überwachtes Lernverfahren für Feedforward Netze ist Backpropagation of Error. Backpropagation of Error ist ein gradientenbasiertes

iteratives Verfahren, bei dem der Fehler des Netzwerkes, d.h. die Abweichung der aktuell erzielten (o_i) und der gewünschten also korrekten Ausgabe (t_i) minimiert wird. Als normiertes Fehlermaß wird oft der mittlere quadratische Fehler (vgl. Formel 6) genutzt [Kr2013, S.59f, S.89ff; Ze1997, S.106ff; BSS2013, S.7].

Formel 6: Mittlerer quadratischer Fehler

$$Err = \sum_{i=0}^{n} 0.5 * (t_i - o_i)^2$$

wobei n die Anzahl der Outputneuronen darstellt

Abhängig des Gradienten sowie der eingestellten Lernrate (Schrittweite der Veränderung) werden ausgehend von der Ausgabeschicht rückwärts die Gewichte aller Verbindungen der aktuellen Schicht zu der Vorhergehenden angepasst. Diese Anpassungen können Online oder Offline erfolgen, d.h. nach Präsentation eines jeden Trainingsdatensatzes (Online) oder nach der Präsentation aller Trainingsdatensätze (Offline). Meist werden die Trainingsdaten dem Netz mehrmals präsentiert, dabei wird jeweils eine komplette Präsentation aller Trainingsdaten als Epoche bezeichnet. Für ein erstes Lernen des Netzes wird meist der Offline-Modus genutzt, da dieser oft performanter ist, der Online-Modus eröffnet hingegen Möglichkeiten, die Netze zur Laufzeit weiter zu adaptieren, d.h. durch Präsentation neuer Trainingsdaten das Verhalten permanent zu optimieren [Kr2013, S.59f, S.89ff; Ze1997, S.106ff; BSS2013, S.7f].

Ein unerwünschter Effekt, der im Zuge des überwachten Lernens auftreten kann, ist das Überlernen des Netzes (overfitting), wobei die präsentierten Trainingsdaten eher auswendig gelernt werden und keine echte Approximation mehr stattfindet. Problem des Überlernens ist, dass nicht in den Trainingsdaten enthaltene Eingangsdatenkombinationen ggf. zu schlechten Ergebnissen führen. Aus diesem Grund wird üblicherweise vor dem eigentlichen Trainingsprozess aus der Menge der Trainingsdaten eine Teilmenge (Validierungsdaten) extrahiert, die allein zur Ergebnisvalidierung und nicht zum Training genutzt wird. Der Trainingsprozess wird als beendet betrachtet, wenn eine definierte Fehlerrate für alle Trainingsdaten unterschritten wird, wenn der Validierungsfehler entgegen den Trainingsfehler steigt ("Überlernen") oder wenn die Maximalzahl der definierten Trainingsepochen erreicht ist (vgl. Abbildung 32) [Kr2003, S.59ff; BSS2013, S.7f].

Auf eine detaillierte mathematische Betrachtung und Herleitung des Backpropagation of Error und alternativer Lernverfahren, die Bestimmung der Lernrate usw. wird an dieser Stelle verzichtet und auf entsprechende Literatur verwiesen [Kr2013, S.89ff; Ze1997, S.106ff].

3.2.2.3 Approximation der Steuerungsfunktionen durch KNN

Im Fall der Approximation von Steuerungsregeln werden sämtliche Merkmale der relevanten Aufträge, d.h. aller zum Entscheidungszeitpunkt im Puffer befindlichen Aufträge, als Input des KNN interpretiert, der Output des Netzes stellt die Priorisierung dar. Die Zahl der Inputneuronen entspricht dabei der Zahl der Auftragsmerkmale plus einen Belegungsflag multipliziert mit der Maximalzahl der Pufferplätze. Outputneuronen sind entsprechend der Maximalzahl der Pufferplätze vorzusehen. Wie in den Ausführungen zum MLP bereits aufgeführt, gilt für alle Ein- und Ausgaben eines Neurons und somit des gesamten Netzes, dass die Eingangswerte zwischen -1 und 1 liegen sollen. Um dies auch im Fall der Approximation der Steuerungsregeln zu gewährleisten, müssen die Auftragsmerkmalswerte entsprechend normalisiert werden. Die konkrete Rechenregel für die Normalisierung hängt aber direkt vom Merkmal ab. Für Merkmale wie beispielsweise die Bearbeitungszeit (vgl. Formel 7), Rüstzeit (vgl. Formel 8) oder den Deckungsbeitrag (vgl. Formel 9) usw. werden die maximal vorkommenden Merkmalswerte, z.B. die maximale Bearbeitungszeit, als +1 interpretiert, alle anderen Werte werden linear auf den Bereich -1 bis +1 skaliert [BS2011, S.97f; BSS2013, S.6-8].

Formel 7: Normierung der Bearbeitungszeit (PT)

$$normalize(PT_i) = \frac{PT_i \cdot 2}{max(PT)} - 1$$

Formel 8: Normierung der Rüstzeit (ST)

$$normalize(ST_i) = \frac{ST_i \cdot 2}{max(ST)} - 1$$

Formel 9: Normierung des Deckungsbeitrags (CM)

$$normalize(CM_i) = \frac{CM_i \cdot 2}{max(CM)} - 1$$

Für zeitabhängige Merkmale, d.h. Merkmale deren Interpretation vom aktuellen Zeitpunkt abhängt, wie der Liefertermin (vgl. Formel 10), müssen zusätzliche Rechenschritte, wie beispielsweise das Bilden eines Deltas zwischen Zeitpunkt und aktueller Zeit erfolgen [BS2011, S.98; BSS2013, S.7].

Formel 10: Normierung des geplanten Liefertermins (DD)

$$normalize(DD_i) = \frac{(DD_i - t_{current}) \cdot 2}{max(\Delta DD_i)} - 1$$

wobei: ΔDD die Differenz zum Liefertermin darstellt

In allen Berechnungsvorschriften ist ersichtlich, dass Maximalwerte benötigt werden. Gerade für Merkmale wie (Delta-) Liefertermin ist aber trotz Analyse der BDE-Daten nicht sichergestellt, dass die ermittelten Werte im Einsatz nicht übertroffen werden.

Das Überschreiten der Maximalausprägung muss nicht gesondert abgefangen werden und führt in der Regel nur zu leichten Genauigkeitsverlusten, da durch die Verarbeitung in den sigmoiden Aktivierungsfunktionen alle Werte über 1 und unter -1 gleich den Maximalwerten gehandhabt werden.

Die für das überwachte Lernen nötigen Trainingsdaten umfassen neben den Eingangsdaten des KNN auch die Ausgangsdaten. Diese sind für die Trainingsphase ebenfalls zu normieren, jedes Ausgabeneuron entspricht in der hier festgelegten Netzarchitektur einem Pufferplatz und kann als Flag interpretiert werden, d.h. es nimmt den Wert eins oder null an, wobei mit eins genau der Pufferplatz gekennzeichnet ist, der den Auftrag mit der höchsten Priorität enthält (vgl. Formel 11).

Formel 11: Normierung Ausgangsdaten für die Lernphase des KNN

$$o_i = \begin{cases} 1, & wenn\ der\ Auftrag\ als\ nächster\ zu\ priorisieren\ ist \\ 0, & sonst \end{cases}$$

In der Nutzungsphase gilt, dass nach der Präsentation der Eingangsdaten in normierter Form der Auftrag zu priorisieren ist, dessen dem Pufferplatz zugeordneten Ausgabeneuron den höchsten Wert angenommen hat. Die Qualität der Ergebnisse hängt dabei von den Trainingsdaten sowie den Netz- und Lernparametern ab.

3.2.2.4 Szenario- und Architekturbeschreibung

Beim Einsatz des KNN Ansatzes in einem Realszenario kommen für die Bereitstellung der Trainings- und Validierungsdaten vor allem BDE Systeme in Betracht. Im Rahmen dieser Arbeit wird aus Gründen der Verfügbarkeit, der Validierbarkeit usw. auf eine emulierte [16] Produktion zurückgegriffen, die realitätsnahe BDE-Daten erzeugt (vgl. Abbildung 27). Dieses Vorgehen erlaubt auch gezielte Tests, indem Steuerungsregeln und sonstige Systemparameter im Emulations- /Simulationsmodell eingestellt werden und der Lernerfolg dieser im KNN beobachtet und somit validiert werden kann [BS2011, S.99f; BSS2013, S.8ff].

Alle folgenden Analysen nutzen ein einfaches Testszenario, in dem eine Arbeitsstation und ein dazugehöriger Puffer enthalten sind ("single server model"). Der Puffer weist eine Kapazität von fünf auf, jeder Auftrag der nicht sofort auf der Bearbeitungsstation bearbeitet werden kann, wird im Puffer abgelegt. Die Basisparameter der fünf Auftragstypen können Tabelle 10 entnommen werden. Im Rahmen der Untersuchungen wurde mit verschiedenen Auslastungen experimentiert, wobei für die grundlegenden und vergleichenden Untersuchungen eine 70%ige Auslastung des Systems eingestellt

[16] Eine Emulation ist die Nachahmung des bekannten Verhaltens eines Systems durch ein anderes [WIKI2013a]

wurde, konkret wurde eine neg. exponentiell verteilte Zwischenankunftszeit mit einem Erwartungswert von 4:00 Minuten eingestellt [BS2011, S.97; BSS2013, S.9].

Tabelle 10: Auftragsparameter [BSS2013, S.10]

Auftrags -typ	Bearbeitungszeit (PT) [in min]	Liefer-termin	Deckungsbeitrag [in Geldeinheiten]	Rüstzeit [in s]	Häufigkeit [in %]
A	2:00.0000	Freigabetermin + (2 ± 0,5) *PT	5	30	20
B	3:00.0000		30	15	20
C	4:00.0000		70	90	20
D	5:00.0000		100	30	20
E	2:30.0000		60	60	20

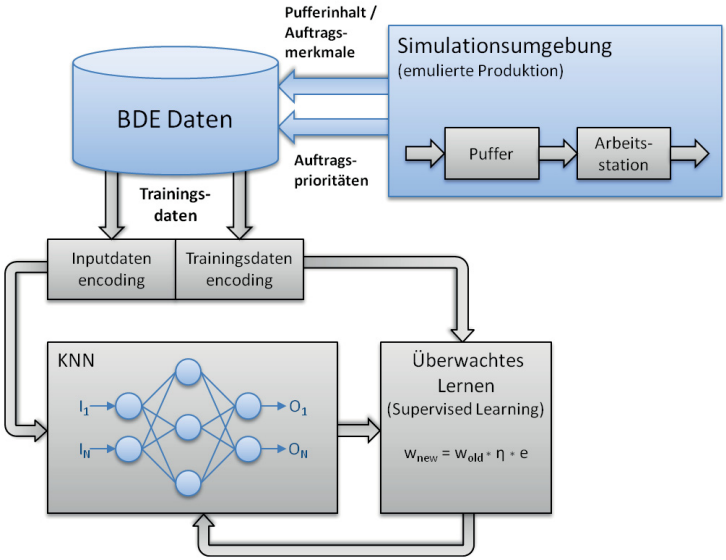


Abbildung 27: Schematische Darstellung des Trainingsprozesses unter Nutzung der Emulation (in Anlehnung an [BSS2013, S.8])

Die Implementierung des Emulationsmodells erfolgte beispielhaft in Siemens Plant Simulation (Abbildung 28), ist aber in jedem kommerziellen Simulator analog implementierbar. Trainingsdaten werden im Simulationsmodell zu jedem Zeitpunkt erzeugt, wenn eine Entnahmeentscheidung im Puffer getroffen wird, d.h. es werden alle Parameter aller im Puffer befindlichen Aufträge normiert und gespeichert

(Tabelle TSnapShot), nicht belegte Pufferplätze werden mit dem Belegungsflag 0 gekennzeichnet sowie sämtliche zugehörige Parameter auf -1 gesetzt. In einer weiteren Zeile der Tabelle wird das Ergebnis der Entscheidung, d.h. welcher Auftrag entnommen wurde dokumentiert indem der Wert auf 1 (Rest 0) gesetzt wird [BSS2013, S.8].

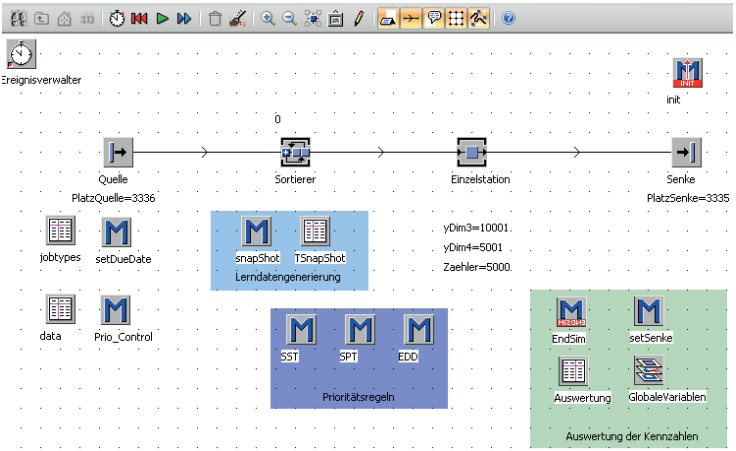


Abbildung 28: Plant Simulation Emulationsmodell für den Trainingsprozess [BSS2013, S.8]

Da in den meisten Simulationsumgebungen keine KNN Funktionalitäten vorgesehen sind, müssen die so erzeugten Datensätze an ein entsprechendes KNN Framework übergeben werden. Prinzipiell ist der hier vorgestellte Ansatz bzgl. des KNN Frameworks neutral, einzig das externe Ansprechen muss technisch realisierbar sein. Ein erster Prototyp basierte auf der freien Fast Artifical Neural Network Libary[17] (FANN) [BS2011, S.99], die hier vorgestellten Ergebnisse basieren sämtlich auf der etablierteren und mächtigeren kommerziellen MATLAB Neuronale Netze Toolbox[18] [BSS2013, S.11f].

Nach dem Training des Netzes soll dieses aus der Simulation heraus genutzt werden, in diesem Fall wird keine konkrete Regel im Simulator implementiert, sondern bei jeder Entscheidung wird entsprechend ein trainiertes KNN genutzt. Analog der Trainingsdatenerzeugung werden zunächst sämtliche Parameter normiert, bevor diese dem Netz präsentiert werden und es ein entsprechendes Ergebnis liefert (vgl. Abbildung 29). Die Ergebnisse des KNN werden im Simulator interpretiert, indem das Ausgabeneuron mit dem Maximalwert ermittelt wird, anschließend wird der Auftrag priorisiert, auf dem der diesem Neuron zugeordnete Pufferplatz liegt [BSS2013, S.9f].

[17] Fast Artificial Neural Network Library; http://leenissen.dk/fann/wp/
[18] Mathworks MATLAB Neural Network Toolbox - Create, train, and simulate neural networks http://www.mathworks.de/products/neural-network/

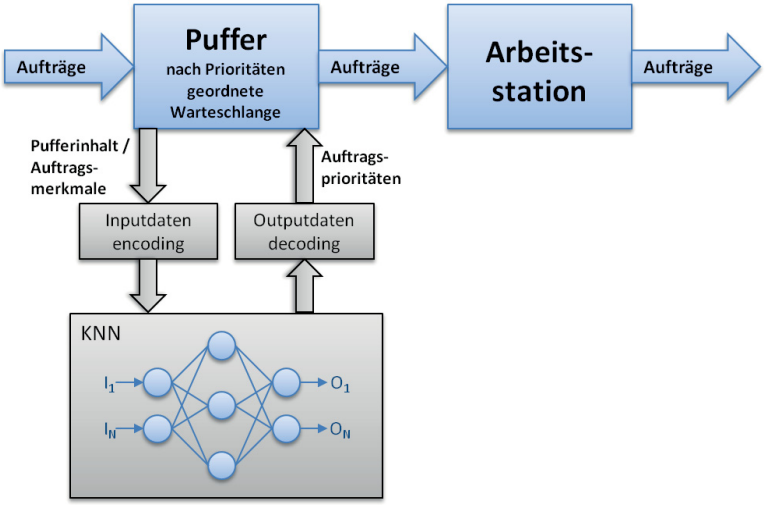


Abbildung 29:Schema des KNN Testszenarios (in Anlehnung an [BSS2013,S.9])

Die gesamte prototypische Implementierung in Plant Simulation (vgl. Abbildung 30) sowie der MATLAB Neuronale Netze Toolbox nutzt als Schnittstelle eine aus Plant Simulation aufrufbare Wrapper-Bibliothek [BSS2013, S.8f].

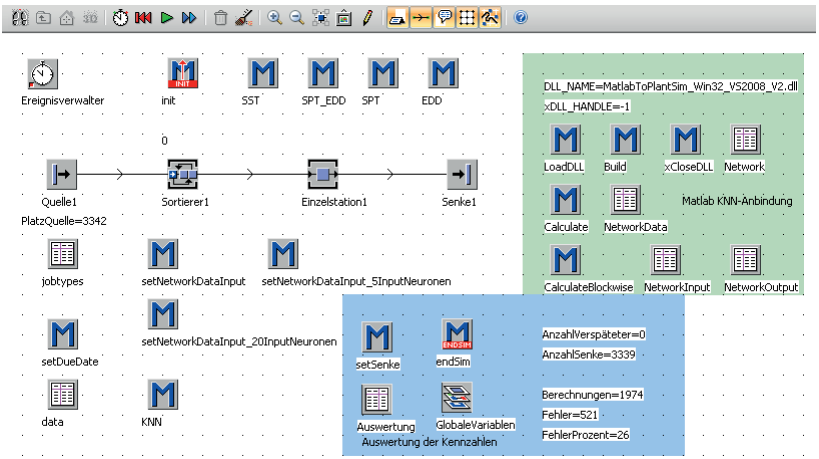


Abbildung 30: Plant Simulation Modell des einfachen Testszenarios [BSS2013, S.9]

Die Wrapper-Bibliothek stellt Funktionen sowohl für das Trainieren und Verwalten als auch für das Berechnen der Ausgaben bereit, siehe Tabelle 11.

Tabelle 11: Übersicht über die Funktionen des Plant Simulation - MATLAB Wrappers (in Anlehnung an [BSS2013, S.11])

Funktionsname	Beschreibung
initEnvironment	Initialisiert die MATLAB-Engine und das interne KNN Management
trainNetworkBatch	Konfiguriert und trainiert ein neues Netzwerk im Offlinemode, basierend auf übergebenden Parametern und Trainingsdaten.
trainNetworkOnline	Konfiguriert und trainiert ein neues Netzwerk im Onlinemode, basierend auf übergebenden Parametern und Trainingsdaten.
calcNetworkOutput	Berechnet das Netzergebnis auf Basis aktueller Eingabedaten
loadNetwork	Lädt ein bestehendes Netzwerk (Matlab-Project).

3.2.2.5 Voruntersuchungen zur Bestimmung der Netzparametern

Im Zuge einer Reihe von Voruntersuchungen wurden gute Einstellungen bzgl. der Netztopologie und der Lernparameter ermittelt, d.h. es wurden solche Parameter gesucht, die zum einen eine gute Approximation aber auch einen zügig konvergierenden Lernprozess gewährleisten. Zudem dienten diese Voruntersuchungen als Machbarkeitsstudie und zur Validierung der Normierungsregeln. Für die Voruntersuchungen wurden Trainingsdaten erzeugt, wobei im Emulationsmodell die Regel "kürzeste Bearbeitungszeit" (engl. shortest processing time - SPT) eingestellt wurde. Im Anschluss an das Lernen eines entsprechenden Netzes wurden Test mittels Simulation durchgeführt. In Rahmen dieser Tests wurden für KKN basierte und regelbasierte Reihenfolgesteuerung bei gleichem Auftragsmix Kennziffern, wie die Durchlaufzeit usw., ermittelt und verglichen. Zudem wurde jede KNN Entscheidung analysiert und dokumentiert, vor allem Art und Anzahl der "Fehlentscheidungen" waren im Fokus der Betrachtungen. In diesem Kontext wird von einer "Fehlentscheidungen" ausgegangen, wenn die hinterlegte Regel (in diesem Fall "kürzeste Bearbeitungszeit") bei der aktuellen Datenlage abweichend zum KNN entschieden hätte. Zunächst wurde dem Netz allein die normierte Bearbeitungszeit präsentiert (5 Pufferplätze = 5 Eingabe und 5 Ausgabeneuronen), diese einfache Regel konnte bereits ab einer verdeckten Schicht approximiert werden. Im nächsten Schritt wurde das Belegungsflag aufgenommen, d.h. je Pufferplatz waren nun zwei Eingabeneuronen vorhanden, diese Maßnahme erhöhte die Genauigkeit der Ergebnisse nochmals. In weiteren Schritten wurden zusätzliche, für die Entscheidung irrelevante Parameter der Aufträge, z.B. Liefertermin, Rüstzeit usw. dem Netz präsentiert. Ergebnis dieser Versuche war, dass für

diese erweiterten Fälle zumindest zwei verdeckte Schichten benötigt werden, wobei dann eine wiederum recht gute Regelapproximation gewährleistet wird (vgl. Tabelle 12). Klar ersichtlich ist, dass die Zahl der betrachteten Eingangsneuronen, die proportional zu der Zahl der betrachteten Auftragsparameter ist, keinen signifikanten Einfluss auf die Ergebnisqualität hat [BS2011, S.99; BSS2013, S.10f].

Tabelle 12: Vergleich der Regel "kürzeste Bearbeitungszeit" mit KNN-Ergebnissen bei unterschiedlicher Eingangsneuronennanzahl (in Anlehnung an [BSS2013, S.11])

Reihenfolge Ermittlung mittels	Durchschnittliche Durchlaufzeit in [min]	Durchschnittliche Verspätung in [min]	Maximale Termin-abweichung	mittlere quadratische Fehler des KNN	Anzahl von Fehl-entscheidungen in 40 Stunden	Anteil der Fehl-entscheidungen
Regel-implementierung	10:26	8:44	-5:29	-	-	-
KNN (5 Eingangs-neuronen)	10:13	9:03	-3:16	0,0029	26	6,52 %
KNN (10 Eingangs-neuronen)	10:08	9:07	-3:10	0,0061	18	4,51 %
KNN (15 Eingangs-neuronen)	10:11	8:49	-3:13	0,0028	25	6,27 %
KNN (20 Eingangs-neuronen)	10:12	9:03	-3:14	0,0050	22	5,51 %

In allen folgenden Experimenten werden Netze mit drei verdeckten Schichten genutzt (vgl. Abbildung 31), eine weitere Erhöhung der Schichtanzahl zeigte keine positiven Auswirkungen auf die Ergebnisse oder den Lernprozess [BS2011, S.99; BSS2013, S.10f].

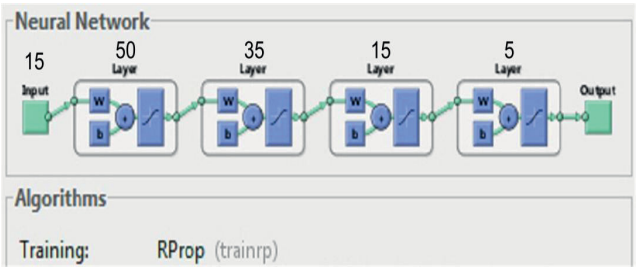


Abbildung 31: Beispielkonfiguration eines KNN in der MATLAB Neuronale Netze Toolbox [BSS2013, S.11]

Aufbauend auf diese grundlegenden Erkenntnisse wurden Tests mit weiteren Netz- und Lernparameteren durchgeführt. So wurde mit verschiedenen Lernraten, die für die Ergebnisgenauigkeit und die Konvergenz des Lernprozess entscheidend sind (siehe [Kr2013, S.94f]), mit der maximalen Epochenanzahl (vgl.Abbildung 32), welche u.a. den Effekt des Auswendiglernens beeinflusst, und der Zahl der Neuronen in den versteckten Schichten, welche sämtliche Performancewerte beeinflusst, experimentiert, die Ergebnisse sind Tabelle 13 zu entnehmen [BS2011, S.99; BSS2013, S.10f].

Tabelle 13: Lern- und Netzwerkparameter (in Anlehnung an [BSS2013, S.10])

Parameter	Min.	Max.	Bester Wert
Lernrate	0,05	0,4	0,1
Epochenanzahl	50	10.000	200
Zahl versteckte Schichten	0	5	3
Anzahl Neuronen in der ersten versteckten Schicht	5	100	50
Anzahl Neuronen in der zweiten versteckten Schicht	5	50	35
Anzahl Neuronen in der dritten versteckten Schicht	5	50	5

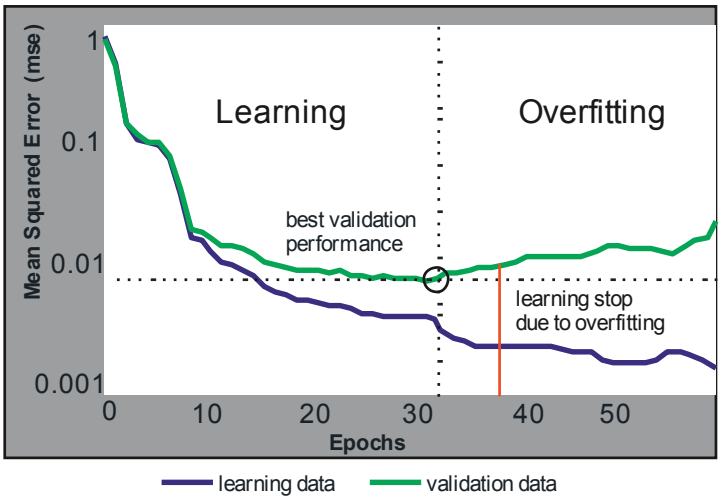


Abbildung 32: Beispielhafter Verlauf des mittleren quadratischen Fehlers im KNN Lernprozess; Bei Epoche 38 wurde der Prozess zur Vermeidung von "Überlernen" beendet [BSS2013, S.10]

Die hier so ermittelten Einstellungen sind Basis aller weiteren Experimente. In weiteren Voruntersuchungen mit anderen Prioritätsregeln, z.B. "frühester Liefertermin" fiel auf, dass kontinuierliche Wertebereiche bei Eingangsparametern, z.B. Delta-Liefertermin, den Lernaufwand erhöhen und ggf. leicht schlechtere Ergebnisqualität mit sich bringen. Diesem Fakt kann z.B. durch Erhöhung der Zahl von Trainingsdatensätzen, Epochen usw. entgegen gewirkt werden.

3.2.2.6 Experimente

Nach der Bestimmung geeigneter Netz- und Lernparameter mittels Voruntersuchungen wurde in umfangreichen Experimenten untersucht, ob und wie gut verschiedene Regeln und Regelkombinationen durch KNN approximiert werden können. Dazu wurde das bereits beschriebene Testszenario erweitert und wie in den Voruntersuchungen, sowohl Kennziffern des Systems als auch Art und Anzahl von "Fehlentscheidungen" betrachtet. Für die Untersuchungen wurden die lokalen Regeln: FIFO, "kürzeste Bearbeitungszeit" (engl. shortest processing time - SPT), "frühester Liefertermin" (engl. earliest due date-EDD), "kürzeste Rüstzeit" (engl. shortest setup time - SST), "höchster Deckungsbeitrag" (engl. highest contribution margin - CM), Schlupfzeitregel (engl. Slack) [19], der multiplikativen Verknüpfung der Schlupfzeit und der "kürzeste Bearbeitungszeit" Regel (vgl. Formel 12), sowie eine Multi-Level Prioritätsregel mit den Teilregeln "kürzeste Bearbeitungszeit" und "frühester Liefertermin" (vgl. Formel 13) genutzt [BS2011, S.99; BSS2013, S.12].

Formel 12: Multiplikative Regel; Schlupfzeit * "kürzeste Bearbeitungszeit"

$$P_i = \left(0.5 * \frac{1}{(t_{dd\,i} - t_{now}) - t_{pt\,i}}\right) + \left(0.5 * \frac{1}{t_{dd\,i}}\right)$$

Formel 13: Multi-Level Prioritätsregel; "kürzeste Bearbeitungszeit" | "frühester Liefertermin"

$$P_i = \frac{1}{t_{pt\,i}}$$
$$\forall P_i = Max(P): \; P_i = P_i + \frac{1}{t_{dd\,i}}$$

[19] Bei Anwendung der Schlupfzeitregel (engl. Slack) wird der Auftrag Priorisiert der bei dem die Differenz zwischen verbleibender Zeit bis zum Liefertermin (Delta-Liefertermin)und der Bearbeitungszeit minimal ist (vgl. [Ne2007, S.694].

Die Ergebnisse sind in Tabelle 14 wiedergegeben alle Simulationsläufe wurden jeweils mit den gleichen Eingangsdaten über einen Zeitraum von 40 Stunden durchgeführt.

Tabelle 14: Vergleich zwischen KNN und regelbasierter Reihenfolgesteuerung (simulatorintern) für verschiedene Prioritätsregeln (in Anlehnung an [BSS2013, S.12])

Reihenfolge Ermittlung mittels	Durchschnittliche Durchlaufzeit in [min]	Durchschnittliche Verspätung in [min]	Maximale Termin-abweichung in [min]	Anzahl von Fehl-entscheidungen in 40 Stunden	Anteil der Fehl-entscheidungen in %
SPT (KNN)	10:16	09:57	-03:38	158	5,83
SPT (simulatorintern)	10:26	08:44	-05:29		
FIFO (KNN)	10:20	08:15	-03:38	145	5,37
FIFO (simulatorintern)	10:23	08:32	-05:08		
EDD (KNN)	10:09	07:04	-03:32	103	3,78
EDD (simulatorintern)	10:16	07:37	-05:17		
SST (KNN)	10:27	09:41	-07:08	192	7,68
SST (simulatorintern)	11:14	09:17	-08:43		
CM (KNN)	11:44	14:26	-05:04	122	4,54
CM (simulatorintern)	12:26	12:12	-07:29		
Slack (KNN)	10:47	08:15	-05:49	191	7,07
Slack (simulatorintern)	11:30	08:52	-06:32		
SPT+EDD (KNN)	10:48	08:13	-05:49	153	5,65
SPT+EDD (simulatorintern)	11:15	08:38	-06:17		
Slack*SPT (KNN)	10:51	08:18	-05:54	236	8,75
Slack*SPT (simulatorintern)	11:15	08:38	-06:17		

In den folgenden Abbildungen sind für einige ausgewählte Regeln die Ergebnisse grafisch aufgearbeitet (vgl. Abbildung 33, Abbildung 34, Abbildung 35 und Abbildung 36).

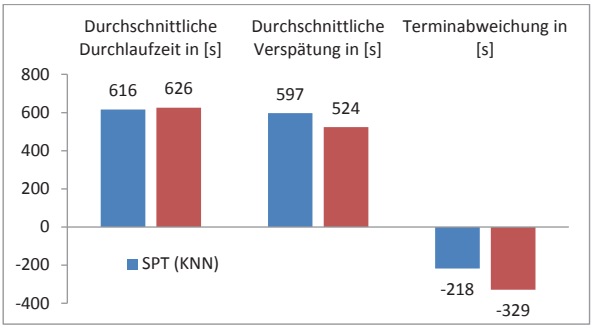


Abbildung 33: Evaluation der Regel "kürzeste Bearbeitungszeit" (SPT) und deren KNN-Approximation (in Anlehnung an [BSS2013, S.13])

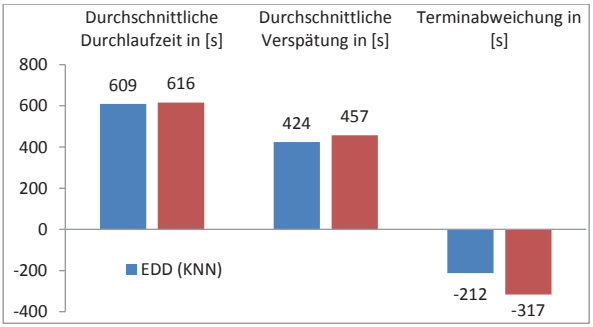


Abbildung 34: Evaluation der Regel "frühester Liefertermin" (EDD) und deren KNN-Approximation (in Anlehnung an [BSS2013, S.13])

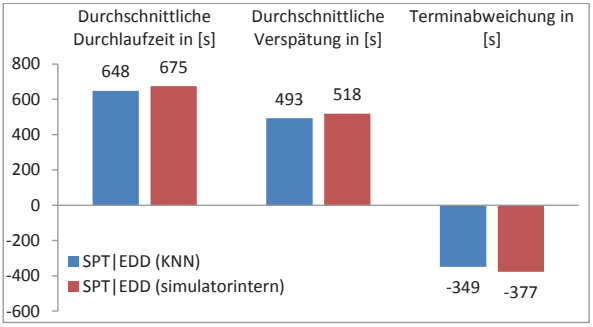


Abbildung 35: Evaluation der Multi-Level Prioritätsregel; "kürzeste Bearbeitungszeit" und "frühester Liefertermin" (SPT|EDD) und deren KNN-Approximation (in Anlehnung an [BSS2013, S.13])

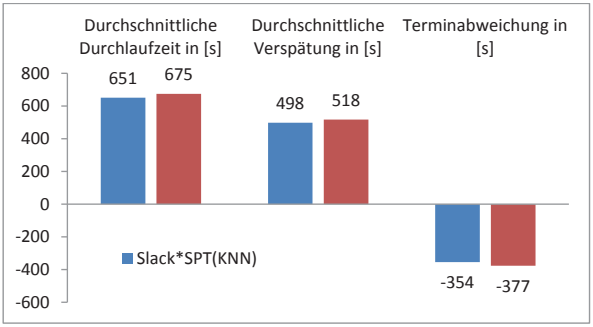


Abbildung 36: Evaluation der multiplikativen Regel Schlupfzeit und "kürzeste Bearbeitungszeit" (Slack*SPT) und deren KNN-Approximation (in Anlehnung an [BSS2013, S. 13])

Die Ergebnisse wurden von Bambl [Ba2011] durch eine Vielzahl von Versuchsreihen erfolgreich verifiziert.

3.2.2.7 Fazit

In allen Tests (vgl. Tabelle 12) wurde eine gute Abbildung der Regel durch das KNN erreicht. Die Fehlerrate betrug selbst bei Kombinationen aus ungünstigen Regeln, d.h. Regeln welche mit kontinuierlichen Eingangsdaten arbeiten, z.B. EDD, immer deutlich unter 9%, in den weniger komplexen Fällen war die Fehlerrate mit um die 5 % nochmals deutlich niedriger. Trotz der auftretenden Fehlerrate stellt das KNN eine signifikante Verbesserung zu der Annahme einer falschen Strategie, z.B. FIFO, dar. So zeigten Versuche auf Basis des gleichen Szenarios und mit gleichen Eingangsdaten, dass wenn fälschlicherweise FIFO statt "kürzeste Bearbeitungszeit" angenommen wird, eine Fehlentscheidungsrate von ca. 35 %, also mehr als 6 mal höher als bei der Anwendung des KNN auftritt. Die Performance des Ansatzes ist durchweg als ausreichend zu bezeichnen, die Antwortzeiten des KNN bei der Nutzung im Testszenario waren so gering, dass sie in der Testumgebung (Plant Simulation) nicht sinnvoll messbar waren. Der Lernprozess der der Nutzung vorrausgeht, benötigt in allen Versuchen wenige Minuten (<5) und stellt, da nur einmalig durchzuführen, keine Limitierung dar.

Somit kann klar gesagt werden, dass der hier skizzierte Ansatz durchaus geeignet ist, aus bestehenden Daten, z.B. BDE-Daten, (Reihenfolge-)Steuerungsregeln zu approximieren und diese auch in der Simulation verfügbar zu machen und somit einen Beitrag zur Abbildung von Strategien im Kontext der Modellgenerierung zu leisten.

Zukünftige Untersuchungen müssen zeigen, ob ein Einsatz in der Praxis sinnvoll zu gestalten ist. So sind Untersuchungen in komplexeren Szenarien mit unterschiedlichen Puffergrößen mehreren Maschinen, weiteren Auftragsparametern, mit globalen Prioritätsregeln usw. sinnvoll. Ebenso muss die Performance des Ansatzes, vor allem z.B.

bei steigender Eingangsneuronenanzahl, weiter untersucht werden. Auch die Einbindung von KNN in andere Simulatoren, z.B. SLX, die ggf. bessere Analysen ermöglichen und/oder höhere Anforderungen bzgl. der Performance der Entscheidungen aufweisen, ist eine mögliche Entwicklungsrichtung. Zudem sind weitere Untersuchungen auf Seite des KNN denkbar, so sind Versuche mit anderen Netztopologien (z.B. Jordan, Elman oder Hopfield Netze; siehe [Kr2013, S.127]), Lernverfahren und -parametern, Normierungsvorschriften für Eingangsdaten oder die Nutzung von Verfahren der Neuroevolution untersuchungsrelevant. Weiterhin sind die Möglichkeiten, die das Onlinelernen des Netzes mit sich bringen nicht vollständig untersucht, so ist ein spannendes Forschungsfeld beispielsweise wie gut sich die Netze an sich ändernde Regeln des Realsystems anpassen können, d.h. wie gut sie adaptieren. Hierbei ist vor allem die Geschwindigkeit und Genauigkeit der Adaption in geeigneter Weise zu analysieren. Schließlich kann der KNN-basierte Ansatz auch für weitere Strategiecluster, z.B. für die Ressourcenbelegung, angepasst werden.

Für den Einsatz in der Modellgenierung oder auch webbasierten Simulation müssen einmal gelernte Netze zudem als Teil der Eingangsdaten abbildbar sein, dazu sind Möglichkeiten der Speicherung der Netze in einem möglichst neutralen Format zu untersuchen. Ziel muss es sein KNN so abzubilden, dass eine Integration in z.B. CMSD möglich ist. XML basierte Ansätze, z.B. die Extensible Markup Language For Artificial Neural Networks [XA2013], scheinen hierbei erfolgversprechend.

3.2.3 Abbildung dynamischen Verhaltens im Framework

Nach der allgemeinen Diskussion der Möglichkeiten zur Abbildung dynamischen Verhaltens (vgl. Abschnitt 3.2.2) wird im Folgenden die aktuelle Implementierung im Framework zur Simulationsmodellgenerierung, -nutzung, -adaption und -validierung angerissen.

Prinzipiell lassen sich alle 3 Klassen von Ansätzen (vgl. Abschnitt 3.2.2; kombinierbare Basisregeln, systemneutrale Modellierung und Approximation der Regeln) zur Abbildung dynamischer Regeln in das Framework integrieren. Alle Varianten weisen je nach Anwendungsfokus verschiedene Stärken und Schwächen auf, die in den entsprechenden Abschnitten bereits angerissen wurden. So ist zu erkennen, dass nicht alle Ansätze in allen Fällen gleich gute Ergebnisse liefern und sich mitunter im anfallenden Aufwand erheblich unterscheiden können. Des Weiteren sind spezifische Voraussetzungen einzelner Ansätze gegeben. So ist beispielsweise die Approximation bzw. auch das automatische Erkennen von Basisregeln bzw. Basisregelkombinationen nur möglich wenn entsprechende BDE Daten vorliegen. Für die Integration des KNN Ansatzes beispielsweise ist zudem der offene Punkt der neutral integrierbaren Abbildung der Netze zu klären. Für das hier nicht im Detail betrachtete Modellieren von

Verhalten in einer systemneutralen Darstellung zur Codegenerierung sowie das Definieren von Basisregelkombinationen sind mitunter keine BDE Daten erforderlich, wohl aber ist mitunter mit erheblichem zusätzlichen ggf. manuellem Aufwand bei der Konzeptmodellerstellung zu rechnen. Weiterhin ist für die systemneutrale Modellierung des Verhaltens ggf. entsprechende Werkzeugunterstützung sicherzustellen, dies kann die Entwicklung bzw. die Integration von entsprechenden Methoden und Softwarekomponenten beinhalten.

Allgemein müssen alle Ansätze in der Best Practice Lösung vorgesehen sein, um eine entsprechende Implementierung in allen abgeleiteten Modellgeneratoren sicherzustellen. In Anbetracht der geschilderten Gründe wird aktuell nur die Definition von Basisregeln unterstützt. Dies stellt aber keine Einschränkung des Konzeptes im Allgemeinen sondern allein eine Konzession an den Umfang der Arbeit dar. Zudem zeigte die Validierung im Feldexperiment (vgl. Abschnitt 4.2) die gute Praxiseignung des Ansatzes.

Für Details der Umsetzung in der CMSD Interpretation sei auf die folgenden Abschnitte verwiesen. So sind Regeln zur Reihenfolgeplanung (decisionRule, vgl. Tabelle 19) in Puffern vor Bearbeitungsstationen bereits essentieller Teil des Anwendungsfalls 1 (vgl. Abschnitt 3.3.1), in Anwendungsfall 4 (vgl. Abschnitt 3.3.4) werden Regeln zum Routing auf parallele Maschinen (RoutingRule; vgl. Tabelle 29) aufgenommen. Das Setzen der Regeln kann innerhalb der in Abschnitt 3.4.3 beschriebenen Komponente, dem CMSDWebfrontend, erfolgen.

Eine Erweiterung der bestehenden Interpretation ist wie im Vorgehensmodell (vgl. Abschnitt 3.1) vorgesehen jederzeit möglich.

3.3 Interpretation des CMSD Standards

Wie bereits vorab diskutiert, ist CMSD zwar ein durchaus gut geeigneter Standard zum Zweck des Datenaustausches im Rahmen der Simulation, im Speziellen der Modellgenerierung, -initialisierung, -adaption und Ergebnisauswertung, bedarf aber wie im Abschnitt 2.3.4 dargelegt im Detail einer Interpretation und in einigen wenigen Fällen einer Erweiterung. Die genaue Interpretation ebenso wie die Definition von Pflichtfeldern, sinnvollen Kardinalitäten und zusätzlich nötigen Properties im Sinne einer Beispiel bzw. Best Practice Lösung stellt ein wesentliches Ergebnis dieser Arbeit dar. Darüber hinaus wurden kleinere Schwächen des Standards erkannt und Vorschläge zur Behebung, in beispielsweise einer späteren Version, aufgeführt. Diese Erweiterungen sind mit der aktuellen Version des Standards nicht konform, daher werden sie im Folgenden gesondert gekennzeichnet. Ein Beispiel für solch eine Erweiterung wäre die Ergänzung eines Aufzählungsdatentyps um zusätzlich erlaubte Werte, z.B. Einfügen des Wertes "setup" in die Liste erlaubter "ResourceStatus".

Es wird eine umfangreiche Interpretation des CMSD Standards mit dem Fokus der Abbildung der Werkstattfertigung vorgestellt, welcher als ein für typische "Produktionsszenarien" untereinander kompatibler erweiterbarer Interpretationsvorschlag bzw. Referenzimplementierung dient. Eine für jedes denkbare Produktionssystem abschließende vollständige Lösung wird als nicht realisierbar angesehen. Die im Folgenden vorgestellte Interpretation hat hierbei das Ziel so gestaltet zu sein, dass weitere Aspekte jederzeit durch den Autor selbst bzw. auch andere Anwender des CMSD Standards ergänzt werden können, womit sukzessive eine Komplettierung, Erweiterung bzw. Abrundung dieser Referenzimplementierung möglich ist.

In den folgenden Abschnitten wird eine Auswahl der aus Sicht des Autors essentiellen Anwendungsfälle aufgeführt. Die Anwendungsfälle sind allesamt im Produktionskontext, im Speziellen im Kontext einer Werkstattfertigung zu sehen und weitestgehend aufeinander aufbauend, d.h. wenn nicht anders vermerkt, wird in jedem Fall die Funktionalität des vorhergehenden Falls erweitert bzw. verfeinert. Die Auswahl der Werkstattfertigung als Grundlage aller Beispiele beruht auf der Annahme, dass die Werkstattfertigung als Produktionsprinzip, vor allem in KMU's, als typisch anzusehen ist und somit potentiell ein großer Bedarf besteht. Zum anderen ist die Werkstattfertigung an sich so komplex, dass andere Produktionssysteme, z.B. die Fließfertigung, im Kontext dieser Arbeit als Vereinfachung anzusehen sind.

Alle Anwendungsfälle sind explizit einfach gehalten um wesentliche Konzepte zu verdeutlichen und sind nicht direkt von real existierenden Produktionssystemen abgeleitet. Die Erkenntnisse können aber nichts desto trotz analog als auf reale

Szenarien übertragbar angesehen werden. Reale Systeme, selbst in KMU's, sind zwar meist deutlich komplexer, als die hier genutzten Szenarios, dies beruht aber meist nicht auf komplexeren Fertigungskonzepten sondern vor allem auf einer höheren Zahl an Systementitäten.

Eine realitätsnahe Implementierung eines Produktionssystems mittels des CMSD Standards wird im Zuge des 4. Kapitels, Validierung des Konzeptes/Frameworks, ausführlich betrachtet und kritisch analysiert. Auszüge der Interpretation unterschiedlicher Entwicklungsstadien mit unterschiedlichem Fokus wurden auf namhaften Konferenzen, wie beispielsweise der ASIM Fachtagung und der Winter Simulation Conference, veröffentlicht [Be2011; Be2012; BSS2010a; BSS2011a; BSS2011b; BSS2012]. Die vollständigen CMSD XML Beispieldateien sind Anhang D zu entnehmen.

Wie bereits im Abschnitt 2.3 dargelegt, bietet der CMSD Standard die Möglichkeit zur strukturierten Erweiterung (mittels Properties), aus Gründen der Kompatibilität werden im Rahmen der Interpretation so wenige dieser Erweiterungen wie möglich genutzt. Zudem sind prinzipiell alle Erweiterungen so angelegt, dass ein Funktionieren des Konzepts (und der Prototypen vgl. Abschnitt 3.4.2) auch ohne diese gewährleistet wird, allein die Genauigkeit des Modells wird unter Umständen verringert, d.h. es wird eine stärkere Abstraktion vorgenommen.

3.3.1 Anwendungsfall 1 - einfache 3x3 Werkstattfertigung

Anwendungsfallbeschreibung:

Als Grundszenario aller folgenden beispielhaften Anwendungsfälle zur Veranschaulichung der CMSD Interpretation, dient eine einfache fiktive Werkstattfertigung mit 3 Maschinen und 3 verschiedenen Produkten (3x3), wobei jedes Endprodukt über einen eigenen Arbeitsplan/Prozessplan definiert ist, d.h. jeweils eine eigene technologisch bedingte Bearbeitungsreihenfolge (Reihenfolge der für die Bearbeitung benötigte Bearbeitungsstation, z.B. Maschinen, Handarbeitsplätze) sowie spezifische Bearbeitungszeiten und benötigte Rüstzustände an jeder dieser Station aufweist (siehe Abbildung 37, Tabelle 15, Tabelle 16 und Tabelle 17).

Arbeits-/Prozesspläne:

technischer Produktionsprozess (= Endprodukt)	Maschinenreihenfolge (Bearbeitungszeiten)
Produktionsprozess A (Produkt A)	A (10:00; RZ1) → B (13:40; RZ1) → C (11:20; RZ1)
Produktionsprozess B (Produkt B)	B (20:00; RZ2) → C (15:10; RZ2) → A (12:00; RZ1)
Produktionsprozess C(Produkt C)	C (13:30; RZ3) → A (10:45; RZ1) → B (18:00; RZ3)

Produktionsprogramm:

ID	Produkt	Liefertermin	…
A1	A	1.2.2014	
A2	B	3.1.2014	
…			

Rüstmatrix:

RZi	RZ1	RZ2	…
RZ1	-	5:20	
RZ2	5:20	-	
RZ3	3:00	5:50	-
…			

Produktionssystem:

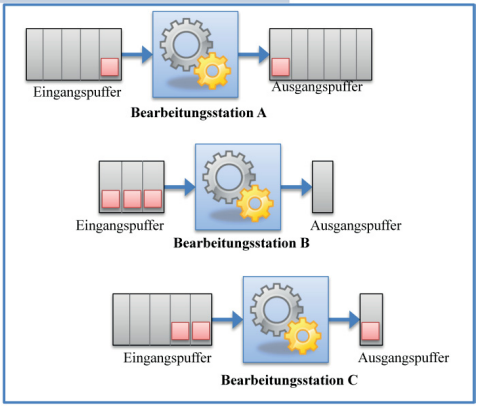


Abbildung 37: Schematische Darstellung der einfachen 3x3 Werkstattfertigung (RZ i= Rüstzustand i)

Tabelle 15: Parameter der Bearbeitungsstationen - Anwendungsfall 1

Parameter	Bearbeitungs-station A	Bearbeitungs-station B	Bearbeitungs-station C
Eingangspuffer-kapazität	5	3	5
Ausgangspuffer-kapazität	6	1	1
X - Koordinate (Ein-; Ausgangspuffer)	250 (200; 300)	300 (250; 350)	350 (300; 400)
Y - Koordinate (Ein-; Ausgangspuffer)	100 (100; 100)	200 (200; 200)	300 (300; 300)

Tabelle 16: Rüstmatrix am Beispiel der Bearbeitungsstation A - Anwendungsfall 1

Von \ Nach (in Minuten)	Rüstzustand (RZ) 1	Rüstzustand (RZ) 2	Rüstzustand (RZ) 3
Rüstzustand (RZ) 1	-	5:20	4:50
Rüstzustand (RZ) 1	5:20	-	4:50
Rüstzustand (RZ) 1	3:00	5:50	-

Tabelle 17: Arbeitspläne - Anwendungsfall 1

Produkt	Nummer des Prozessschritts	Bearbeitungsstation	Benötigter Rüstzustand	Operationszeit (in Minuten)
Produkt A	1	A	RZ 1	10:00
	2	B	RZ 1	13:40
	3	C	RZ 1	11:20
Produkt B	1	B	RZ 2	20:00
	2	C	RZ 2	15:10
	3	A	RZ 1	12:00
Produkt C	1	C	RZ 3	13:30
	2	A	RZ 1	10:45
	3	B	RZ 3	18:00

In diesem einfachen Anwendungsszenario wird zunächst nur von noch nicht begonnen 60 Aufträgen ausgegangen (vgl. Tabelle 18). Auch werden abgerüstete und ungestörte Maschinen angenommen. Somit ist keine Initialisierung (vgl. Abschnitt 2.2.2) nötig.

Tabelle 18: Produktionsprogramm (Auszug) - Anwendungsfall 1

ID des Fertigungsauftrags	Herzustellendes Produkt	Gewünschter Liefertermin	Frühester Freigabetermin
Job_PP1_1	Produkt A	01.02.2014	02.01.2014
Job_PP2_1	Produkt B	03.01.2014	02.01.2014
Job_PP3_1	Produkt C	03.01.2014	02.01.2014
Job_PP3_2	Produkt C	06.02.2014	16.01.2014

Für den Werkskalender wird ein einfacher Zwei-Schichtbetrieb angenommen. Die Frühschicht beginnt hierbei um 5 Uhr, wird von Pausen um 7 Uhr (15 Minuten) sowie 11 Uhr (30 Minuten) unterbrochen und endet 13 Uhr. Die Spätschicht beginnt um 14 Uhr,

wird von Pausen um 16 Uhr (15 Minuten) sowie 20 Uhr (30 Minuten) unterbrochen und endet 22 Uhr (vgl. Anhang E - Auszug aus einer Kalenderdarstellung in CMSD XML). Hierbei wird die Annahme getroffen, dass alle Bearbeitungsstationen als immobile Objekte zu jeder Zeit verfügbar sind.

Modellierungspattern zur Abbildung von Bearbeitungsstationen

Eine grundlegende Annahme, die getroffen wurde, ist dass alle Bearbeitungsstationen ein gemeinsames Modellierungspattern [20] aufweisen, dies gilt auch für sämtliche weitere Spezialisierungen, wenn nicht explizit anders definiert. In diesem Modellierungspattern wird eine Bearbeitungsstation immer durch drei Subelemente (vgl. Abbildung 38) gekennzeichnet:

1. die Bearbeitungsstation selbst,
2. einen Eingangspuffer sowie
3. einen Ausgangspuffer.

Abbildung 38: Allgemeines Modellierungspattern für Bearbeitungsstationen

Praktisch sind in den meisten realen Systemen Puffer vor und/oder nach einer Bearbeitungsstation vorhanden, diese sind aber selten explizit in betrieblichen IT Systemen, z.B. ERP-Systemen hinterlegt, jedoch spielen sie aber gerade für die Steuerung innerhalb der Simulation sowie für die Initialisierung des Systems eine entscheidende Rolle. Durch das Modellierungspattern wird der Simulationsnutzer zum einen für die Puffer sensibilisiert, zum anderen helfen die Default-Werte der Eigenschaften der Puffer sogar die Modellqualität zu verbessern. Alle Puffereigenschaften sind jederzeit zu parametrieren, wenn entsprechende Informationen vorliegen. Im Fall, dass keine Ein- bzw. Ausgangspuffer im gesamten System oder an einzelnen Bearbeitungsstationen gewollt oder in der Realität vorhanden

[20] Unter Modellierungspattern werden bewährte Lösungsschablonen für wiederkehrende Modellierungsprobleme verstanden [in Anlehnung an WIKI2012b]

sind (z.B. bei einer Fließfertigung), kann die Pufferkapazität entsprechend auf 0 gesetzt werden. Das Parametrieren mit 0 Kapazitäten ermöglicht, dass auch solche Systeme problemlos abbildbar sind und somit keine Einschränkung des Ansatzes vorliegt.

Der Bearbeitungsstation sind die bereits genannten Parameter wie Bearbeitungszeit und Rüstzustand für einen bestimmten Bearbeitungs-/Prozessschritt hinterlegt. Für beide Puffer sind zusätzliche Parameter bezüglich der Kapazität, d.h. welche Menge an Objekten maximal aufgenommen werden kann und der Reihenfolgestrategie/Steuerungsstrategien (vgl. Abschnitt 2.1.7) hinterlegt.

Die Nutzung dieses Modellierungspatterns schränkt den Ansatz nicht ein und weist eine Reihe von Vorteilen für die Modellgenerierung und -initialisierung auf. So kann aus Sicht des bedienungstheoretischen Grundkonzepts [21] (Warteschlangentheorie) der Eingangspuffer des Patterns dem Wartebereich gleichgesetzt werden.

Abbildung in CMSD:

Für das beschriebene einfache Anwendungsszenario (einfache 3x3 Werkstattfertigung) werden neben den Elementen zur Strukturierung des CMSD Dokuments (z.B. DataSection) und Elementen zur Beschreibung von Metadaten (z.B. UnitDefaults), auf die hier nicht nochmals eingegangen werden soll (vgl. Abbildung 15) grundlegende Elemente aus allen Paketen (vgl. Abbildung 16) des CMSD Information Models benötigt.

Die Anzahl der zwingend benötigten Klassen des CMSD Information Models ist bereits für das überschaubare Anwendungsszenario recht hoch. Zudem ist eine sehr hohe Verknüpfung der Elemente untereinander zu beobachten. In den folgenden Ausführungen und Abbildungen werden zur Wahrung der Übersichtlichkeit nur die zwingend benötigten und einige wenige empfehlenswerte Attribute und Klassen dargestellt, für alle anderen Parameter sei an dieser Stelle wieder auf die Dokumentation des Standards selbst verwiesen [SISO2012; SISO2012b]. Die Nutzung von weiteren, hier nicht aufgeführten Parametern kann je nach Anwendungsfall aber durchaus sinnvoll sein.

[21] Ein Wartesystem besteht üblicherweise aus einem Bedienbereich, in dem ein oder mehrere Aufträge bearbeitet werden, und einem Wartebereich, in dem eintreffende Aufträge bei Belegung des Bedienbereichs auf dessen Verfügbarkeit warten [WIKI2013b].

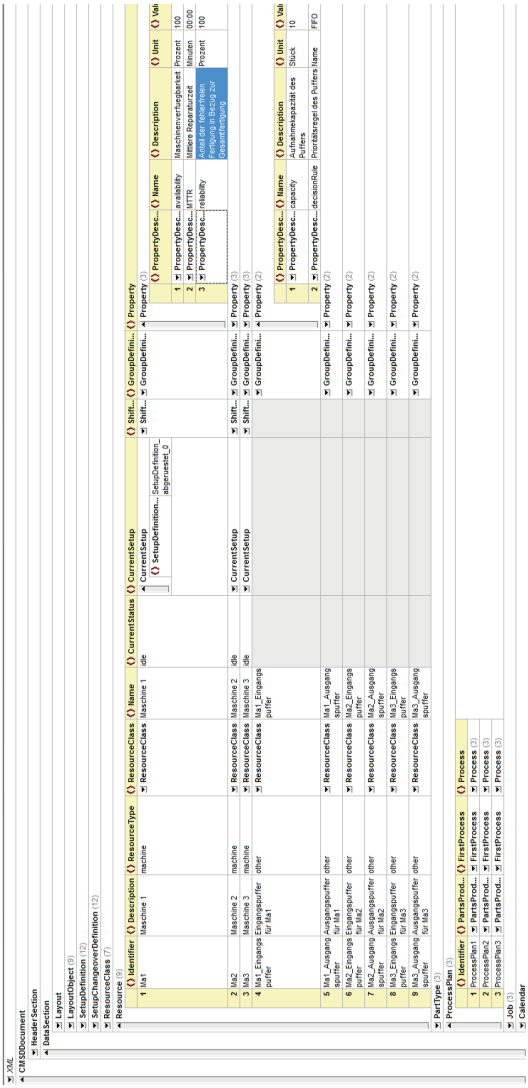


Abbildung 39: Gridview [22] des "einfache 3x3 Werkstattfertigung" CMSD XML Dokuments; inklusive benutzerdefinierten Properties

[22] Gridviews präsentieren Daten in verschachtelten Tabellen, in diesen Fall wurde der XML Editor Altova XMLSpy 2011 [Al2012] zur Visualisierung genutzt

Die benötigten statischen Daten, d.h. die technischen und Organisationsdaten (vgl. Abschnitt 2.1.6; Abbildung 5), sind durch Klassen des Ressource Information und des Layout Package sowie der Klasse Calender des Production Planning Package komplett abbildbar (vgl. Abbildung 39). In Abbildung 40 ist ein stark vereinfachtes Abbild der benötigten Entitäten sowie deren Beziehungen skizziert, in Tabelle 19 ist eine detailliertere Auflistung der entsprechenden Entitäten und Attribute aufgeführt.

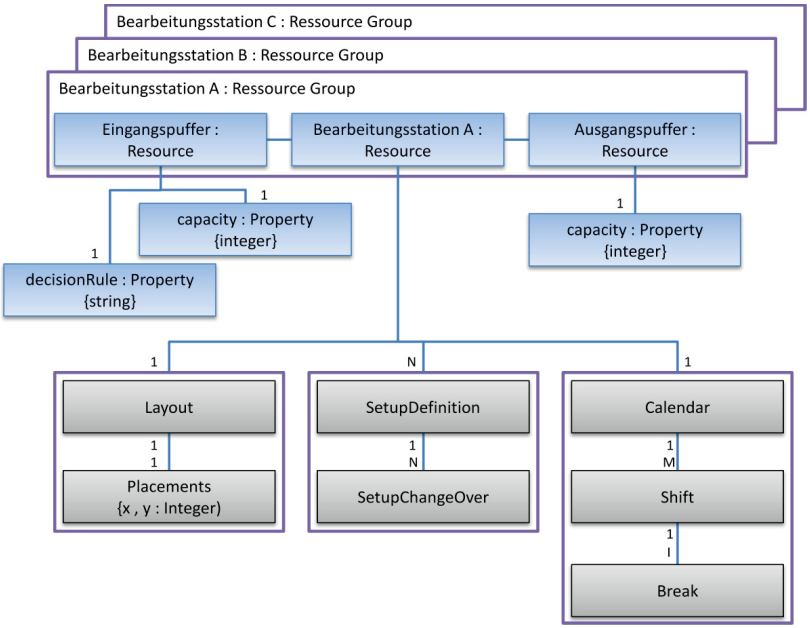


Abbildung 40: Schematische Darstellung der statischen CMSD Entitäten für Anwendungsfall 1

Die Realisierung des beschriebenen Modellierungspatterns erfolgt über die Definition einer entsprechenden Ressourcengruppe (ResourceGroupInformation). Beispielhaft ist die Definition der Bearbeitungsstation A als CMSD XML in Abbildung 41 ersichtlich, für die vollständige CMSD Datei sei auf Anhang E verwiesen.

```
...
<Resource>
    <Identifier>Ma1</Identifier>
    <Description>Station A1</Description>
    <ResourceType>machine</ResourceType>
    <ResourceClass>
        <ResourceClassIdentifier>Ma_simple</ResourceClassIdentifier>
    </ResourceClass>
    <Name>Beabreitungsstation A</Name>
    <CurrentStatus>idle</CurrentStatus>
    <CurrentSetup>
        <SetupDefinitionIdentifier>SetupDefinition_abgeruestet_0</SetupDefinitionIdentifier>
    </CurrentSetup>
    <ShiftAssignment>
        <CalendarIdentifier>Calender_2012</CalendarIdentifier>
        <ShiftIdentifier>SchichtFrueh_Werktags</ShiftIdentifier>
    </ShiftAssignment>
    <GroupDefinition>
        <ResourceGroupMember>
            <ResourceIdentifier>Ma1</ResourceIdentifier>
        </ResourceGroupMember>
        <ResourceGroupMember>
            <ResourceIdentifier>Ma1_Eingangspuffer</ResourceIdentifier>
        </ResourceGroupMember>
        <ResourceGroupMember>
            <ResourceIdentifier>Ma1_Ausgangspuffer</ResourceIdentifier>
        </ResourceGroupMember>
        <Connection>
            <Identifier>Connection_0</Identifier>
            <FromResource>
                <ResourceIdentifier>Ma1_Eingangspuffer</ResourceIdentifier>
            </FromResource>
        </Connection>
        <Connection>
            <Identifier>Connection_1</Identifier>
            <ToResource>
                <ResourceIdentifier>Ma1_Ausgangspuffer</ResourceIdentifier>
            </ToResource>
        </Connection>
    </GroupDefinition>
</Resource>
...
```

Abbildung 41: Definition der Bearbeitungsstation A - Auszug der CMSD Datei Anwendungsfall 1

Tabelle 19: Statische CMSD Entitäten und deren minimale Attribute des Anwendungsfalls 1 - einfaches 3x3 Werkstattszenario

CMSD Entität	Relevante Attribute (Kardinalität)			Datentyp (Wertebereiche)	Beschreibung / Interpretation
Resource {Resource Type ≠ employee}	Identifier (1)			Identifier	Eineindeutige ID der Ressource, Pflichtfeld nach CMSD Standard
	Description (0..1)			String	Kurzbeschreibung der Ressource
	ResourceTyp (1)			resourceType {Ein-/Ausgangspuffer = "other" (oder "buffer"*)} {Bearbeitungsstation = "machine" "station"}	Typisierung der Ressource
	Name (0..1)			String	Optionaler "sprechender" Name
	capacity (Property) {nur für Puffer (1)}			Integer	Maximalkapazität des Puffers; -1 = ∞
	decisionRule (Property) {nur für Eingangspuffer (1)}			decisionRules **	Angabe der Reihenfolgeregel
	ResourceGroupInformation (1)	Resource GroupMember (3..N)		ResourceReference	Liste aller zusammengehöriger Ressourcen --> Modellierungspattern
		Connection (1..N)	Identifier (1)	Identifier	Eineindeutige ID, Standardpflichtfeld
			From Resource {nur für Bearbeitungs-station und Ausgangs-puffer} (0..N)	ResourceReference	Vorgängerbeziehung einer Entität
			ToResource {nur für Bearbeitungs-station und Eingangs-puffer} (0..N)	ResourceReference	Nachfolgerbeziehung einer Entität

Setup Definition	Identifier (1)		Identifier	Eineindeutige ID des Rüstzustandes, Standardpflichtfeld
	Description (0..1)		String	Kurzbeschreibung des Rüstzustandes
	SetupResource (1)		ResourceReference {nur Bearbeitungs-stationen}	Referenz auf die Bearbeitungsstation für die dieser Rüst-zustand erlaubt ist
	Name (0..1)		String	Optionaler "sprechender" Name der Rüstzustandes
Setup Change Over	Identifier (1)		Identifier	Eineindeutige ID der Umrüstvorgangsliste, Standardpflichtfeld
	CurrentSetup (1)		SetupDefinition Reference	Ausgangsrüstzustand, d.h. Rüstzustand von dem aus umgerüstet werden soll
	NewSetup (0..N)	SetupDefinition Identifier (1)	SetupDefinition Reference	Zielrüstzustand, d.h. Rüstzustand auf den umgerüstet werden soll
		ChangoverTime (1)	Duration	Für die Umrüstung benötigte Zeit (Wert oder Verteilung und Einheit)
Layout	Identifier (1)		Identifier	Eineindeutige ID eines Objektes des Fertigungslayouts, Standardpflichtfeld
	Description (0..1)		String	Kurzbeschreibung des Layoutobjektes
	AssociatedResource (1)		ResourceReference	Referenz auf eine Resource, die in diesem Layoutobjekt platziert ist
	Boundary (1)		BoundaryDefinition	Definition der Lage und Größe des Layoutobjektes
	Name (0..1)		String	Optionaler "sprechender" Name des Layoutobjektes

	Placement (1)	LayoutElement Identifier (1)		Identifier	Eineindeutige ID eines Objektes des Placements, Standardpflichtfeld; ACHTUNG abweichende Namensgebung!
		Location (1)		Coordinate3D	Koordinaten des Placements, Interpretation als linke obere Ecke des Objektes
Calendar	Identifier (1)			Identifier	Eineindeutige ID des Kalenders, Pflichtfeld nach Standard
	Description (0..1)			String	Kurzbeschreibung des Kalenders
	Shift (1..N)	Identifier (1)		Identifier	Eineindeutige ID der Schicht, Pflichtfeld nach Standard
		Description (0..1)		String	Kurzbeschreibung der Schicht
		StartTime (1)		Time	Schichtbegin
		Duration (1)		ElapsedTimeType	(Netto) Dauer der Schicht (Wert + Einheit)
		ApplicableDay (1..7)		Day	Wochentage für die die Schicht gültig ist
		Break (0..N)	Description (0..1)	String	Optionale Kurzbeschreibung der Pause
			StartTime (1)	Time	Startzeitpunkt der Pause
			Duration (1)	ElapsedTimeType	Dauer der Pause (Wert + Einheit)

* nicht standardkonform, als Erweiterung des Standards durch den Autor empfohlen, siehe Anhang C - Die wichtigsten CMSD Aufzählungsdatentypen

** zusätzlich entworfener Aufzählungsdatentyp, siehe Anhang C - Die wichtigsten CMSD Aufzählungsdatentypen

Die Systemlastdaten (vgl. Abschnitt 2.1.6; Abbildung 5) werden mit Klassen der Packages Production Planning, Production Operation und Part Information abgebildet, siehe Abbildung 42 sowie Tabelle 20. Die Konzepte des Support Package werden in nahezu jeden Teilaspekten benötigt, z.B. Datentypen, Properties, Referenzen usw.

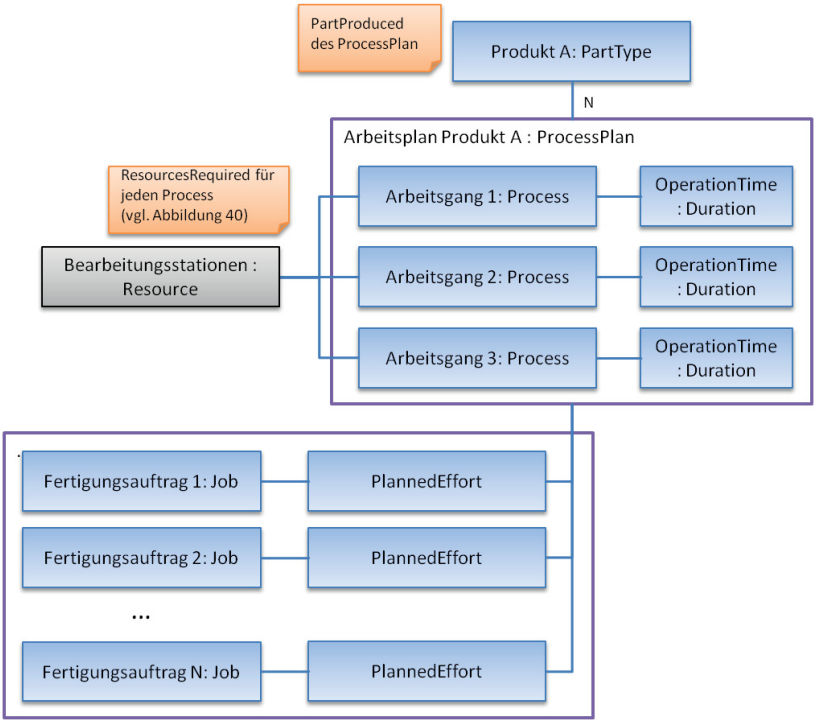


Abbildung 42: Schematische Darstellung der dynamischen CMSD Entitäten für Anwendungsfall 1

Ein Auszug aus der resultierenden CMSD XML Beschreibung des Arbeitsplans für Produkt A sowie eines Auftrags für dieses Produkt kann Abbildung 43 entnommen werden.

```
...
<ProcessPlan>
        <Identifier>ProcessPlan1</Identifier>
        <PartsProduced>
                <Description>Product1</Description>
                <PartType>
                        <PartTypeIdentifier>PartType1</PartTypeIdentifier>
                </PartType>
                <PartQuantity>1</PartQuantity>
        </PartsProduced>
        <FirstProcess>
                <ProcessPlanIdentifier>ProcessPlan1</ProcessPlanIdentifier>
                <ProcessIdentifier>PP1_ProcessA</ProcessIdentifier>
        </FirstProcess>
        <Process>
                <Identifier>PP1_ProcessA</Identifier>
                <Description>Prozess A des Prozessplans 1</Description>
                <ResourcesRequired>
                        <Description>PP1_PA_Ma</Description>
                        <MinimumNumber>1</MinimumNumber>
                        <MaximumNumber>1</MaximumNumber>
                        <Resource>
                                <ResourceIdentifier>Ma1</ResourceIdentifier>
                        </Resource>
                        <AllowableSetup>
                                <SetupDefinitionIdentifier>
                                        Setup_S1_Ma1
                                </SetupDefinitionIdentifier>
                        </AllowableSetup>
                </ResourcesRequired>
                <OperationTime>
                        <Unit>minute</Unit>
                        <Value>30</Value>
                </OperationTime>
        </Process>      ...
</ProcessPlan>      ...
<Job>
    <Identifier>Job_PP1_1</Identifier>
    <Description>Job Prozessplan1 Nr.1</Description>
    <Status>released</Status>
    <UpdateTime>2012-07-13T13:54:30</UpdateTime>
    <PlannedEffort>
        <UpdateTime>2012-07-13T13:54:24</UpdateTime>
        <DueDate>2012-07-13T20:00:00</DueDate>
        <ReleaseDate>2012-07-13T07:00:00</ReleaseDate>
        <StartTime>2000-01-01T00:00:00</StartTime>
        <EndTime>2000-01-01T00:00:00</EndTime>
        <ProcessPlan>
            <ProcessPlanIdentifier>ProcessPlan1</ProcessPlanIdentifier>
        </ProcessPlan>
    </PlannedEffort>
</Job> ...
```

Abbildung 43: Definition des Arbeitsplans A und des Auftrags 1 - Auszug der CMSD Datei Anwendungsfall 1

Tabelle 20: Dynamische CMSD Entitäten und deren minimale Attribute des Anwendungsfalls 1 - einfaches 3x3 Werkstattszenario

CMSD Entität	Relevante Attribute (Kardinalität)			Datentyp (Wertebereiche)	Beschreibung / Interpretation
PartType	Identifier (1)			Identifier	Eineindeutige ID des Bauteil-/Produkttyps, Pflichtfeld nach Standard
	Description (0..1)			String	Kurzbeschreibung des Bauteil-/Produkttyps
	Name (0..1)			String	Optionaler "sprechender" Name des Bauteil-/Produkttyps
Process Plan	Identifier (1)			Identifier	Eineindeutige ID des Arbeitsplans, Pflichtfeld nach Standard
	PartProduced (1)	Description (0..1)		String	Kurzbeschreibung des produzierten Endprodukts des Arbeitsplans
		PartType (1)		PartTypeReference	Referenz auf einen Produkttyp
		PartQuantity (1)		NonNegativeInteger	Anzahl der durch den Arbeitsplan produzierten Endprodukte
	FirstProcess (1)			ProcessReference	Referenz auf den Startteilprozess des Arbeitsplans
	Process (1..N)	Identifier (1)		Identifier	Eineindeutige ID des Teilprozesses, Pflichtfeld nach Standard
		Description (0..1)		String	Kurzbeschreibung des Teilprozesses
		ResourcesRequiered	Description (0..1)	String	Kurzbeschreibung der für den Teilprozess benötigten Ressourcen-kombination

			Resource (1)	ResourceReference {nur Bearbeitungs-stationen}	Referenz auf die für den Teilprozess der benötigten Bearbeitungsstation
			Allowable Setup (1)	SetupDefinition Reference	Referenz auf die für den Teilprozess an der Bearbeitungsstation benötigten Rüstzustand
		OperationTime (1)		Duration	Für den Teilprozess benötigte Bearbeitungszeit (Verteilung oder Wert und Einheit)
Job	Identifier (1)			Identifier	Eineindeutige ID des Fertigungsauftrags, Pflichtfeld nach Standard
	Description (0..1)			String	Kurzbeschreibung des Arbeitsauftrags
	PlandEffortDescription (1..2)	ReleaseDate (1)		TimeStamp	Geplanter Freigabe-termin, d.h. frühester Produktionsstart
		DueDate (0..1)		TimeStamp	Geplanter Liefertermin (z.B. genutzt für Steuerstrategien)
		ProcessPlan (1)		ProcessPlan Reference	Referenz auf den hinterlegten Arbeitsplan des Fertigungsauftrags

Neben der hier vorgestellten Interpretation (vgl. Abbildung 38, Abbildung 40, Abbildung 42, Tabelle 19 und Tabelle 20) ist eine Reihe von abweichenden Lösungen denkbar, so könnte das Modellierungspattern für Bearbeitungstationen ungenutzt bleiben, Properties anders definiert werden oder Attribute des Standards anders gedeutet werden, z.B. könnte statt ReleaseDate auch StartTime als frühester Produktionsbeginn genutzt werden.

3.3.2 Anwendungsfall 2 - einfache 3x3 Werkstattfertigung mit stochastischen Einflüssen, z.B. Störungen

Anwendungsfallbeschreibung:

Eine erste Erweiterung des in Anwendungsfall 1 geschilderten rein deterministischen Grundmodells ist im Einbringen von Zufallselementen zu sehen. Stochastik ist ein entscheidender Punkt für die Wahl der Methode Simulation, da z.B. in rein mathematischen Modellen stochastische Einflüsse schwer oder nicht abbildbar sind. Die Modellierung der Stochastik ist im Zuge der Modellerstellung ein nicht zu unterschätzender Aufwand. Obwohl die überwiegende Zahl der Simulatoren den Nutzer gerade hier gut unterstützt, ist vor allem die Datenerhebung und Bestimmung der Verteilungsfunktion und Parameter teils sehr aufwendig und selten a priori vorhanden.

Stochastische Einflüsse können verschiedenste Simulationsaspekte beeinflussen. Dabei wird zumeist die Dauer, das Auftreten oder der Eintrittszeitpunkt eines bestimmten Zustands (z.B. Verfügbar, In Bearbeitung) beeinflusst. Die typischsten Beispiele hierfür sind stochastische Rüst-, Bearbeitungs- oder Reparaturzeiten, stochastische Verfügbarkeit von Ressourcen, durch Störungen von Maschinen oder Ausfälle von Werkern sowie die stochastische Verteilung von Auftragsdaten (Art, Menge, Termine usw.), zufällig erscheinende Reihenfolge, Routingentscheidungen oder Ausschussquoten innerhalb des modellierten Bearbeitungsprozesses.

Konkret erweitert Anwendungsfall 2 das Basismodell um stochastische Störungen von Bearbeitungsstationen sowie um stochastische Bearbeitungszeiten, weitere stochastische Zeiten z.B. Rüstzeiten sind analog abbildbar und werden aus Gründen der Übersichtlichkeit hier nicht explizit betrachtet.

Das vorab in Anwendungsfall 1 definierte Produktionssystem wird um die Annahme erweitert, dass alle Bearbeitungsstationen (A, B, C) ausfallen also stören können. Weiterhin wird angenommen, dass Daten bzgl. des Auftretens der Störungen ebenso wie benötigte Zeiten zur Behebung dieser bekannt sind (vgl. Tabelle 21).

Tabelle 21: Beispielhaftes Störverhalten - Anwendungsfall 2

Parameter	Bearbeitungs-station A	Bearbeitungs-station B	Bearbeitungs-station C
Verfügbarkeit	80%	90%	100% [23]
Mittlere Reparaturzeit (MTTR)	1:00:00	2:00:00s	--

[23] Wird unterstellt, dass an einer Bearbeitungsstation keinerlei Störungen auftreten können, kann eine Verfügbarkeit von 100% angenommen werden. Eine Angabe zu Reparaturzeiten wird in diesem Fall obsolet.

Zusätzlich werden die Arbeitspläne für alle Produktvarianten auf Bearbeitungsstation A angepasst, indem deterministische durch stochastische Bearbeitungszeiten ersetzt werden (vgl. Tabelle 22), sämtliche weitere Einstellungen bleiben unverändert.

Tabelle 22: Ergänzung stochastischer Bearbeitungszeiten im Arbeitsplan - Anwendungsfall 2

Produkt	Nummer des Prozessschritts	Operationszeit (in Minuten)
Produkt A	1	**Gleichverteilt von 9:00 bis 11:00**
Produkt B	3	**Gleichverteilt von 11:30 bis 12:30**
Produkt C	2	**Normalverteilt; Mittelwert: 10:45; Varianz: 0:45**

Die stochastische Auftragsverteilung, stochastische Wegwahl bei parallelen Maschinen und stochastische Reihenfolgeentscheidungen etc. sind weitere unter Umständen stochastische Aspekte, die in Produktionssystemen von Bedeutung sind. Diese werden am Ende dieses Abschnittes diskutiert, bedürfen aber keiner gesonderten Interpretation und sind nicht Teil des Anwendungsfalls.

Abbildung in CMSD:

Der CMSD Standard bietet im Paket BasicStructures des Support Paketes grundlegende Strukturen zur Definition von Verteilungen [SISO2012a, S.22, 31]. Dieser Mechanismus kann zur Abbildung von verschiedenen stochastischen Effekten in CMSD Modellbeschreibungen eingesetzt werden.

Die Abbildung stochastisch verteilter Bearbeitung und Rüstzeiten wird mittels der im Standard vorgesehenen Möglichkeit realisiert, bei der Angabe von Zeitdauern (Duration) nicht nur einen Absolutwert und dessen Einheit (z.B. Minuten) anzugeben, sondern auch eine beliebige Verteilungsfunktion (Distribution; vgl. Abbildung 44) hinterlegen zu können.

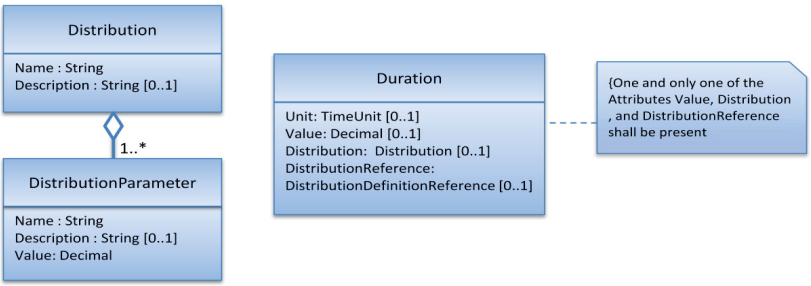


Abbildung 44: Abbildung von Verteilungen im CMSD Standard; UML Notation [SISO2012a, S22]

Allerdings werden im Standard keine Verteilungen explizit definiert, wodurch Interpretationsspielraum bei der Nutzung entsteht. Für den im Rahmen dieser Arbeit diskutierten Anwendungsfokus ist aber eine allgemeingültige und gleiche Explizierung von Verteilungsfunktionen zwingend nötig. Hierzu wird in Tabelle 23 eine Liste typischer Funktionen und deren Abbildung in CMSD vorgestellt. Diese erhebt keinen Anspruch auf Vollständigkeit und kann jederzeit durch weitere Verteilungen erweitert werden. Zu beachten ist nur, dass jede Verteilungsfunktion eineindeutig zu bezeichnen ist (in CMSD genutzter eindeutiger Kurzbezeichner = Name der Distribution) und alle relevanten Parameter vollständig definiert sind (vgl. Tabelle 23 und Abbildung 45).

Tabelle 23: Einige typische Verteilungsfunktionen und Vorschlag für deren Realsierung in CMSD

Bezeichnung der Verteilung (deutsch/ englisch)	CMSD Kurz-bezeichner (Name)	Parametername (Distribution Parameter)	Dichtefunktion
Gleichverteilung / Uniform distribution	uniform	Untere Grenze (a) Obere Grenze (b)	$f(x) = \begin{cases} \frac{1}{b-a}, & a < x \leq b \\ 0, & sonst \end{cases}$
Exponential-verteilung/ Exponential distribution	exponential	Ereignisrate (lamda [λ])	$f(x) = \lambda e^{-\lambda x}$
Normalverteil-ung (Gauß-Verteilung)/ Normal distribution	normal	Mittelwert (mu [μ]) Varianz (sigma[σ])	$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$
Dreiecks-verteilung/ Triangular distribution	triangular	Untere Grenze (a) Häufigster Wert (b) Obere Grenze (c)	$f(x) = \begin{cases} \frac{2\,(x-a)}{(b-a)(c-a)}, & a \leq x \leq b \\ \frac{2\,(c-x)}{(c-b)(c-a)}, & b \leq x \leq c \\ 0, & sonst \end{cases}$
Weibull-verteilung/ Weibull distribution	weibull	Skalierungs-parameter (alpha[α]) Formparameter (beta[β])	$f(x) = \alpha\beta(\alpha x)^{\beta-1} e^{-(\alpha x)^{\beta}}$
Logarithmische Normalerteilung / lognormal distribution	lognorm	Mittelwert (mu [μ]) Varianz (sigma[σ])	$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln(x)-\mu}{\sigma}\right)^2}$

Erlangen Verteilung / erlang distribution	erlang	Mittelwert (mu [μ]) Varianz (sigma[σ])	$f(x) = \begin{cases} \frac{\lambda^n x^{n-1}}{(n-1)!} e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$ mit $\mu = \frac{n}{\lambda}$; $\sigma = \frac{n}{\lambda^2}$
Diskret empirische Verteilung/ discrete empirical distribution	dEmp	Liste von Wertpaaren (Name/Value)	
Kontinuierlich empirische Verteilung/ continuous empirical distribution	cEmp	Liste von Wertpaaren (Name/Value)	

```
<Distribution>
        <Name>normal</Name>
        <Description>Normalverteilung; Mittelwert 0, Varianz 1</Description>
        <DistributionParameter>
                <Name>mu</Name>
                <Description>Mittelwert</Description>
                <Value>10.45</Value>
        </DistributionParameter>
        <DistributionParameter>
                <Name>sigma</Name>
                <Description>Varianz</Description>
                <Value>0.45</Value>
        </DistributionParameter>
</Distribution>
```

Abbildung 45: Beispielhafte Abbildung der Normalverteilung als CMSD konforme XML Datei (Produkt C, Prozessschritt 2)

Die Abbildung von Störungs- und Ausschussprofilen also der Realisierung von Störverhalten von Ressourcen und der Angabe von Ausschussanteilen ist im CMSD Standard nicht direkt vorgesehen. Dies sind einige der wenigen für die Simulation von Produktionssystemen als essentiell anzusehender Aspekte, die keinerlei CMSD Entsprechung aufweisen. Für zukünftige Versionen des Standards ist gerade hier Arbeitsbedarf zu sehen, die im Folgenden dargestellten Lösungen können als direkter Vorschlag zur Standarderweiterung aufgefasst werden. Im Konzept der Modellgenerierung kann schwer auf diese wichtige Informationen verzichtet werden. Die hier präferierte Implementierung nutzt die Möglichkeit, die Ressourcenklasse um Properties zu erweitern. Auf Basis der in vielen Simulatoren üblichen Abbildungsmimik

wird auf eine durchaus ebenso realisierbare Abbildung als Verteilungsfunktion verzichtet und stattdessen die im Folgenden beschriebenen Properties availability (Verfügbarkeit), Mean time to repair (MTTR; mittlere Reparaturzeit) und reliability (Gutteilquote) genutzt (vgl. Tabelle 24).

Nach der VDI Norm 2893 "Auswahl und Bildung von Kennzahlen für die Instanthaltung" Anhang A2 [VDI2893, S. 31] wird die Verfügbarkeit als "Kennzahl, die den Grad der Nutzungsfähigkeit einer Anlage beschreibt" definiert. Die mittlere Reparaturzeit definiert sich als "durchschnittliche Ausfallzeit pro Instandsetzung ", d.h. beinhaltet sowohl die tatsächliche technische Reparaturzeit als auch die Reaktionszeit auf Störungen. In diesem Kontext werden oft weitere Kennzahlen definiert, z.B. Mean Time Between Failures (MTBF) also die Mittlere Zeit zwischen zwei Störungen oder die Mean Time To Failures (MTTF) also die Mittlere Laufzeit einer Anlage ohne Störung (vgl. Abbildung 46).

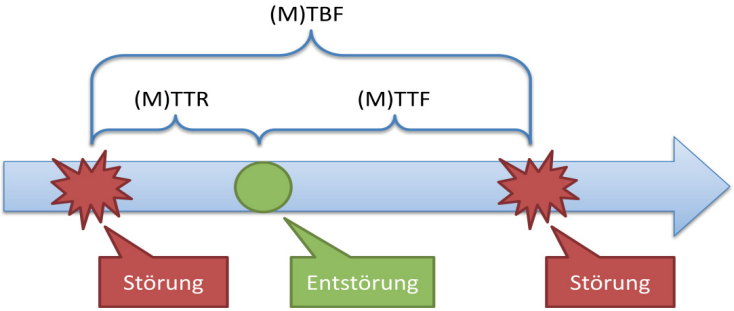


Abbildung 46: Zusammenhang MTTR, MTBF und MTTF [in Anlehnung an Fo2011]

Folgender Zusammenhang besteht zwischen den relevanten Kennzahlen:

(1a) $$MTTR = \frac{MTBF}{Verfügbarkeit} - MTBF$$

oder

(1b) $$Verfügbarkeit = \frac{MTBF}{MTBF+MTTR}$$

und

(2) $$MTBF = MTTR + MTTF$$

wenn

(3) $$MTBF = \frac{\sum TBF}{Anzahl\ (TBF)}$$

(4) $$MTTR = \frac{\sum TTR}{Anzahl\ (TTR)}$$

Es ist ersichtlich, dass bei Angabe von zwei Kennzahlen jeweils die sonstigen berechenbar sind. Da so wenig wie möglich Properties definiert werden sollen, werden allein die Verfügbarkeit (availability) und die MTTR als CMSD Property realisiert (vgl. Tabelle 24). Die durch empirische Forschung gestützte Annahme, die in vielen Simulatoren (z.B. Siemens Plant Simulation) ebenso implementiert ist, ist die, dass zum einen die Zeit zwischen zwei Störungen (MTTF) exponentialverteilt und zum anderen die Reparaturzeit (MTTR) als Erlangen verteilt angenommen wird. Eine weitere aber im Folgenden nicht weiter verfolgte Möglichkeit wäre sowohl die MTTR als auch die MTTF bzw. MTBF als Referenzproperty auf eine Duration abzubilden, womit analog zu Bearbeitungszeiten beliebige Verteilungsfunktionen hinterlegbar wären. In Tabelle 24 sind die für die präferierte Möglichkeit zur Realisierung des Anwendungsfalls 2 zusätzlich zu denen in Tabelle 19 aufgeführten Basisattribute der Bearbeitungsstationen definiert.

Tabelle 24: Erweiterung der CMSD Entitäten Ressource zur Realisierung stochastischer Einflüsse

CMSD Entität	Relevante Attribute (Kardinalität)	Datentyp (Wertebereiche)	Beschreibung / Interpretation
Resource	availability (Property) {nur Bearbeitungstationen} (0..1)	NonNegativeInteger (<=100)	Verfügbarkeit einer Ressource in %
	MTTR (Property) {nur Bearbeitungstationen} (0..1)	Time	Mittlere Reparaturzeit
	reliability (Property) {nur Bearbeitungstationen} (0..1)	NonNegativeInteger (<=100)	Gutteilquote einer Ressource in %

Stochastisches Routing, d.h. eine zufällige Wegwahl im Fall beispielsweise paraller Maschinen und stochastischer Reihenfolgestrategien, d.h. zufällige Priorisierung eines Auftrags, werden als Ausprägung von Entscheidungsregeln (vgl. Abschnitt 3.2) eines Eingangspuffers interpretiert und mittels entsprechender Wertbelegung (= Random) der Properties decisionRule (vgl. Tabelle 19) für Reihenfolgeentscheidungen bzw. routingRule (vgl. Abschnitt 3.3.4, Tabelle 29) für Routingentscheidungen realisiert.

Die gerade im Rahmen der planungsbegleitenden Simulation oft genutzte rein stochastische Modellierung von Auftragsdaten, d.h. keine Angabe einzelner Aufträge sondern der Definition von Verteilungsfunktionen für das Auftreten (Zahl, Art und Zeitpunkte) von Aufträgen einzelner Typen (Produkte) soll in dem hier vorgestellten Konzept nicht als Verteilung in CMSD abgebildet werden. Diese Entscheidung wurde

getroffen, da zum einen der Standard diese Möglichkeit nicht ohne massive Erweiterung vorsieht, zum anderen da die phasenübergreifende Nutzung erschwert werden würde und ggf. nötige Informationen (z.B. zusätzliche Jobparameter) wiederum nicht oder schwer abbildbar wären. Die im Rahmen dieses Konzepts implementierte Lösung ermöglicht einerseits auch Auftragsdaten, die zunächst nur als stochastische Verteilung bekannt sind zu realisieren, ohne andererseits die Prämisse, dass jeder Auftrag auch ein CMSD Job ist zu verletzen. Hierzu werden durch das als Datenquelle genutzte betriebliche IT-System direkt oder durch eine zusätzlich zwischengelagerte Komponente (vgl. Abschnitt 3.4.3) aus den Verteilungen einzelne Jobs generiert und im CMSD eingebunden.

3.3.3 Anwendungsfall 3 - Einbeziehung von Werkern

Anwendungsfallbeschreibung:

Eine weitere typische Erweiterung des Grundszenarios kann in der Einbeziehung von im Produktionsprozess direkt involvierten menschlichen Ressourcen, den Mitarbeitern, Personal oder auch Werker genannt, gesehen werden. Diese zusätzliche Ressource des Produktionsprozesses unterscheidet sich in einigen grundlegenden Eigenschaften von der in Anwendungsfall 1 beschriebenen Bearbeitungsstation.

So wird das Szenario um sechs Werker ergänzt, von denen je drei in der Früh- und drei in der Spätschicht für die Produktion verfügbar sind. Zudem weisen die Werker unterschiedliche Fähigkeiten (engl. Skills) auf. Es erfolgt keine starre Zuordnung einzelner Werker zu einzelnen Aufgaben (bearbeiten, rüsten, reparieren), sondern eine flexible Zuordnung von Werkern zu Aufgaben/Tätigkeiten (z.B. Bearbeitung eines Werkstücks) über den Abgleich aus benötigten und angebotenen Fähigkeiten.

Beispielhaft können die Fähigkeiten zweier Werker Tabelle 25 entnommen werden. Hierbei sind sowohl Fähigkeiten aufgeführt, die für bestimmte Bearbeitungsprozesse benötigt werden, als auch Fähigkeiten, die nicht direkt wertschöpfend sind, wie Fähigkeiten um Rüstzustände von Anlagen zu ändern bzw. Störungen zu beseitigen. Eine vollständige Liste kann Anhang D entnommen werden.

Tabelle 25: Auszug Werkerfähigkeiten - Anwendungsfall 3

Werkername	Schicht	Fähigkeiten
Worker_A	Frühschicht	Skill_A1 (Level 1 und 2), Skill_A2 (Level 1), Skill_B3 (Level 1), Skill_R1 (Level 1), Skill_S1 (Level 1)
Worker_D	Spätschicht	Skill_A1 (Level 1 und 2), Skill_A3,Skill_ Repair_M1, Skill_ Setup _M1

Alle Skills können in verschiedenen Fähigkeitsstufen (Level) auftreten. Diese stellen in der hier vorgeschlagenen Interpretation keine Hierarchie dar, d.h. das Beherrschen eines Levels inkludiert nicht die Fähigkeiten eines anderen vermeintlich niedrigeren Levels. Eine Interpretation gerade als Hierarchie ist aber prinzipiell denkbar und stellte eine für weitere Untersuchungen mögliche Erweiterung dar.

Neben der Definition der Werker und deren Fähigkeiten müssen ebenfalls die Prozesse, als Nachfrager der Fähigkeiten, um entsprechende Eigenschaften erweitert werden. Tabelle 26 zeigt beispielhaft für die Erstellung von Produkt B die für die Arbeitsgänge benötigten Fähigkeiten. Eine vollständige Listung kann wiederum dem Anhang entnommen werden. Bereits in dem Auszug wird deutlich, dass im Gegensatz zu den Bearbeitungsstationen durchaus mehrere Werker gleichzeitig involviert sein können. Dies gilt analog auch für Rüst- und Reparaturprozesse, bei denen ebenso mehrere parallel tätige Werker benötigt werden können.

Tabelle 26: Benötigte Fähigkeiten ausgewählter Bearbeitungsschritte - Anwendungsfall 3

Produkt	Nummer des Prozessschritts	Benötigte Fähigkeiten
Produkt B	1	Skill_A1 (Level 2)
	2	Skill_A1 (Level 1) und Skill_A3 (Level 1)
	3	Skill_B1 (Level2)

Neben den Fähigkeiten zur Bearbeitung von Produkten wird definiert, dass jede Bearbeitungsstation eigene Anforderungen bzgl. Fähigkeiten zur Reparaturdurchführung aufweist, z.B. wird "Skill_ Repair_M2" zur Behebung von Störungen der Bearbeitungsstation 2 benötigt (vgl. Tabelle 27). Wird eine Bearbeitungsstation gerüstet und ist keine abweichende Regelung getroffen, wird angenommen, dass der Werker, welcher die Bearbeitung vornimmt auch für das Rüsten verantwortlich ist, d.h. der oder die Werker werden sowohl über den gesamten Bearbeitungszeitraum als auch während des Rüstens allokiert.

Tabelle 27: Relevante Prozesse an Bearbeitungsstation B - Anwendungsfall 3

Produkt / Prozessschritt	**Benötigte Fähigkeiten**		
	Rüsten	**Bearbeiten**	**Reparieren**
A / 2	Skill_A2Level_1 Skill_A3Level_1		Skill_ Repair_M2
B / 1	Skill_Setup_M2		
C / 3	Skill_Setup_M2	Skill_B1Level_2	

Für Bearbeitungsstation 2 wird beispielhaft unterstellt, dass hier besonders "anspruchsvolle" Rüstvorgänge, die spezielle Fähigkeiten benötigen, auftreten können. So sind verschiedene Varianten für den Fähigkeitsbedarf möglich:

(1) der Standardfall, d.h. eine oder mehrere Fähigkeiten werden sowohl für Bearbeitungsvorgänge als auch für ggf. auftretende Rüstvorgänge benötigt (vgl. Tabelle 27; Zeile: A / 1).
(2) abweichende Rüst- und Bearbeitungsfähigkeiten
    a. voll automatische Bearbeitung, d.h. es werden Werker nur zum Rüsten der Bearbeitungsstation benötigt (vgl. Tabelle 27; Zeile: B / 2)
    b. spezielle Rüstfähigkeit, d.h. es werden spezielle Rüstfähigkeiten benötigt, die ggf. von den Fähigkeiten, die im Rahmen der folgenden Bearbeitung nötig sind, abweichen (vgl. Tabelle 27; Zeile: C / 3)
    c. automatisches Rüsten, d.h. es werden allein zum Bearbeiten Werker benötigt. Das Rüsten erfolgt automatisch oder wird zumindest vernachlässigt.

Abbildung in CMSD:

Zur Abbildung von Werkern in CMSD sind sowohl neue CMSD Entitäten des Resource Information Package, als auch ggf. neue Attribute bestehender Entitäten, z.B. der Bearbeitungsstationen, nötig. Eine grobe schematische Darstellung der beteiligten Elemente kann Abbildung 47 entnommen werden

Da sämtliche Zuordnungen von Werkern zu Tätigkeiten auf Basis von Fähigkeiten erfolgen, ist die Definition eines Fähigkeitenpools mittels der Klasse SkillDefinition nötig.

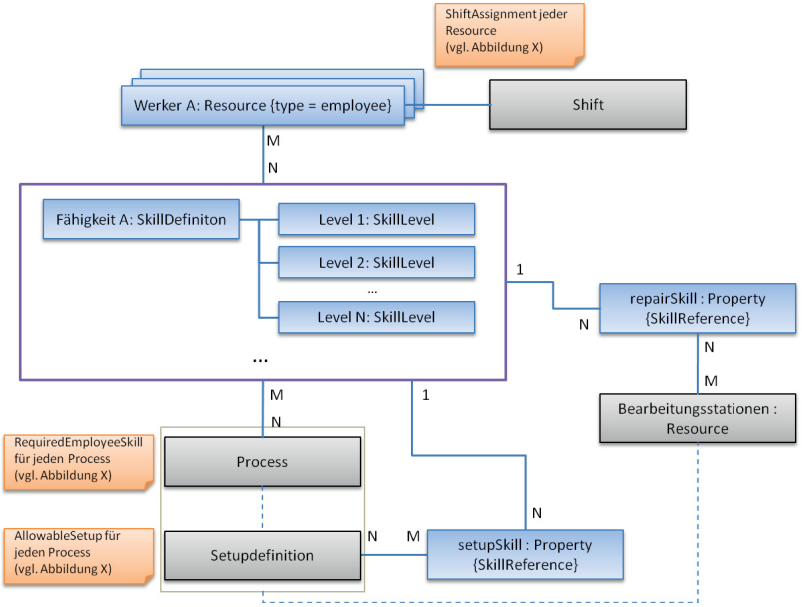


Abbildung 47: Schematische Darstellung der Nutzung von Fähigkeitenbeschreibung für die Werkersteuerung in CMSD

Zur Abbildung der Werker wird, ähnlich der Abbildung von Bearbeitungsstationen oder Puffern, je eine Instanz der CMSD Entität Ressource, aber mit dem RessourceType employee, defininiert. Die benötigten Attribute sind Tabelle 28 und Abbildung 48 zu entnehmen. Den Werkern können hierbei beliebig viele Fähigkeiten des Fähigkeitenpools zugeordnet werden. Zudem unterscheiden sich Werker von sonstigen Ressourcen durch ihren engen Schichtbezug.

Die Abbildung des Fähigkeitenangebots mittels Werkern ist umfänglich ohne jede Erweiterung des Standards möglich.

Der Standardfall, bei dem im Rahmen eines Prozessschrittes eine beliebige Anzahl von Fähigkeiten benötigt wird, kann allein mit dem Erweitern des ResourcesRequired Attributes um die entsprechenden Fähigkeiten abgebildet werden (vgl Tabelle 28, Tabelle 29 und Abbildung 53).

```
...
<SkillDefinition>
    <Identifier>Skill_A1</Identifier>
    <Description>Fähigkeit A1</Description>
    <Name>Fähigkeit A1</Name>
    <SkillLevel>
        <Identifier>Skill_A1Level_1</Identifier>
        <Description>Level 1 Skill A1</Description>
        <Name>Level 1 Skill A1</Name>
    </SkillLevel>
    ...
</SkillDefinition>
...
<Resource>
    <Identifier>Worker_A</Identifier>
    <Description>Werker A</Description>
    <ResourceType>employee</ResourceType>
    ...
    <Name>Anton</Name>
    <CurrentStatus>idle</CurrentStatus>
    <ShiftAssignment>
        <CalendarIdentifier>Calender_2012</CalendarIdentifier>
        <ShiftIdentifier>SchichtFrueh_Werktags</ShiftIdentifier>
    </ShiftAssignment>
    <EmployeeSkill>
        <SkillDefinitionIdentifier>Skill_A1</SkillDefinitionIdentifier>
        <SkillLevelIdentifier>Skill_A1Level_1</SkillLevelIdentifier>
    </EmployeeSkill>
    <EmployeeSkill>
        <SkillDefinitionIdentifier>Skill_A2</SkillDefinitionIdentifier>
        <SkillLevelIdentifier>Skill_A2Level_1</SkillLevelIdentifier>
    </EmployeeSkill>
    ...
</Resource>
...
```

Abbildung 48: Fähigkeiten- und Werkerdefinition - Auszug einer CMSD Datei

Werden abweichende Rüstfähigkeiten benötigt, ist eine Erweiterung der Klasse SetupDefinition um entsprechende SkillReference Properties unumgänglich (vgl. Abbildung 49). Diese Property ordnet jedem Rüstzustand eine Liste der zum Rüsten auf diesen Zustand benötigter Fähigkeiten zu. Eine noch präzisere Zuordnung wäre über die Erweiterung der Rüstprozesse (SetupChangeOver und NewSetup) möglich. Dies scheitert jedoch am CMSD Standard, der keine Properties in der hierfür relevanten Klasse (NewSetup) vorsieht .

Tabelle 28: Zusätzlich benötigte Attribute der Basis CMSD Entitäten zur Abbildung von Werkern - Anwendungsfall 3

CMSD Entität	Relevante Attribute (Kardinalität)			Datentyp (Wertebereich)	Beschreibung / Interpretation
Resource	repairSkill (Property) {nur Bearbeitungsstationen} (0..N)			SkillReference	Benötigte Fähigkeiten zur Reparatur nach einer Störung; eine Fähigkeit entspricht einem Werker
SetupDefinition	setupSkill(Property) (0..N)			SkillReference	Benötigte Fähigkeiten zur Reparatur nach einer Störung; eine Fähigkeit entspricht einem Werker
ProcessPlan	Process	ResourcesRequired	Required EmployeeSkill (0..N)	SkillReference	Benötigte Fähigkeiten zur Durchführung eines Bearbeitungsschrittes; eine RequiredEmployeeSkill entspricht einem Werker

Durch die Erweiterung um nötige Rüstfähigkeiten sind alle vorab beschriebenen Fälle bzgl. Rüst- und Bearbeitungsfähigkeiten hinreichend genau abbildbar.

(1) Standardfall:
Abbildung mittels 2 Varianten möglich
   a. Angabe von Fähigkeiten allein bei der Prozessdefinition (RequiredEmployeeSkills) und keine SetupSkill Poperty
   b. Analoge Angabe von Fähigkeiten sowohl im Prozessschritt (RequiredEmployeeSkill) als auch dem Rüstzustand (SetupSkill Properties)

(2) abweichende Rüst- und Bearbeitungsfähigkeiten
   a. voll automatische Bearbeitung:
      Angabe von Fähigkeiten allein am Rüstzustand (SetupSkill Properties), keine RequiredEmployeeSkill Definition
   b. spezielle Rüstfähigkeiten
      Angabe von ggf. abweichenden Fähigkeiten bei der Prozessdefinition (RequiredEmployeeSkill) und dem Rüstzustand (SetupSkill Properties)
   c. voll automatisches Rüsten:
      Angabe von Fähigkeiten allein bei der Prozessdefinition (RequiredEmployeeSkills) sowie Anlegen einer leeren SetupSkill Property am Rüstzustand.

Die SkillReference Property (repairSkill) der Bearbeitungsstationen rundet die Werkereinbindung ab. Mithilfe dieser Property lassen sich die Fähigkeiten für die Beseitigung von Störungen einer Bearbeitungsstation abbilden (vgl. Tabelle 28, Tabelle 29 und Abbildung 49).

```
...
<SetupDefinition>
    <Identifier>Setup_S1_Ma1</Identifier>
    <Description>Setup 1 Maschine 1</Description>
    <SetupResource>
        <ResourceIdentifier>Ma1</ResourceIdentifier>
    </SetupResource>
    <Name>Setup 1 Bearbeitungsstation A</Name>
    <Property>
        <PropertyDescription>
            <PropertyDescriptionIdentifier>Property_setupSkill</PropertyDescriptionIdentifier>
        </PropertyDescription>
        <Name>setupSkill</Name>
        <Description>Skills die zum Rüsten benötigt werden</Description>
        <SkillReference>
            <SkillDefinitionIdentifier>Skill_Setup_M1</SkillDefinitionIdentifier>
            <SkillLevelIdentifier>Skill_S1_Level1</SkillLevelIdentifier>
        </SkillReference>
    </Property>
</SetupDefinition>
...
<Resource>
    <Property>
        <PropertyDescription>
            <PropertyDescriptionIdentifier>Property_repairSkill</PropertyDescriptionIdentifier>
        </PropertyDescription>
        <Name>repairSkill</Name>
        <Description>Fähigkeit(en) die zur Störungsbeseitigung genutzt wird/werden</Description>
        <SkillReference>
            <SkillDefinitionIdentifier>Skill_Repair_M1</SkillDefinitionIdentifier>
            <SkillLevelIdentifier>Skill_R1_Level1</SkillLevelIdentifier>
        </SkillReference>
    </Property>
</Resource>
...
```

Abbildung 49: Rüst- und Reparaturfähigkeiten - Auszug einer CMSD Datei

Analog der Bearbeitung können Werker in Teilprozesse von Wartungsplänen (Klasse MaintenanceProcess) eingebunden werden.

In allen bisher betrachteten Fällen wird eine Fähigkeit immer durch genau einen Werker erbracht, d.h. jede nachgefragte Fähigkeit entspricht einer Werkeranforderung. Explizit nicht abgedeckt sind also Szenarios in denen mehr als eine angeforderte Fähigkeit von einem Werker erbracht werden kann, z.B. mit dem Ziel so wenig Werker wie möglich zu

allokieren, und Szenarios in dem alternative Fähigkeiten möglich sind. Diese Erweiterungen sind mit Standardmitteln nicht oder nur mit erheblichem Einsatz zusätzlicher Properties direkt abbildbar. Die Einschränkungen können hierbei aber durch geschicktes Modellieren gemildert werden. So können "virtuelle" Fähigkeiten in den Pool aufgenommen werden, die z.B. für eine Gruppe von realen Fähigkeiten stehen und somit Bedarfe an Werkern mit multiplen Fähigkeiten emulieren oder stellvertretend für eine Liste alternativer Fähigkeiten stehen. Dieses Vorgehen ist mit erheblichem Mehraufwand verbunden und bzgl. des Nutzens der erhöhten Detaillierung kritisch zu hinterfragen.

3.3.4 Anwendungsfall 4 - parallele Maschinen

Anwendungsfallbeschreibung:

Bisher wurden allein Prozesspläne betrachtet, bei denen für jeden Prozessschritt / Vorgang (Process) genau eine Bearbeitungsstation definiert wurde. In der Praxis ist dies nicht immer der Fall. So ist durchaus typisch, dass parallele (alternative) Bearbeitungsstationen vorkommen. Ebenso sind Szenarios denkbar, in dem mehrere Ressourcen gleichzeitig benötigt werden.

Konkret wird das Szenario um eine parallel zur Bearbeitungsstation A angelegte Bearbeitungsstation A1 erweitert (vgl. Abbildung 50).

Die alternative Bearbeitungsstation (A1) kann in diesem Szenario in den Produktionsprozessen A und B alternativ zu Station A genutzt werden, benötigt aber eine andere Rüstzustandsdefinition (RZ1a) für die Bearbeitung von Produkt A, sowie abweichende Fähigkeiten der Werker (Skill_A1) im Zuge der Produktion von Produkt B. Des Weiteren benötigt Station A1 die Fähigkeit Skill_Repair_M1a im Rahmen der Störungsbeseitigung und weist lediglich eine Verfügbarkeit von 80% auf.

Die Verteilung der Aufträge auf die beiden Stationen erfolgt möglichst gleichmäßig, d.h. solange keine Störungen auftreten, werden ankommende Aufträge abwechselnd auf den Stationen verteilt. Zu beachten ist hierbei aber die Restriktion, dass Produkt C nicht auf Station A1 zu bearbeiten ist.

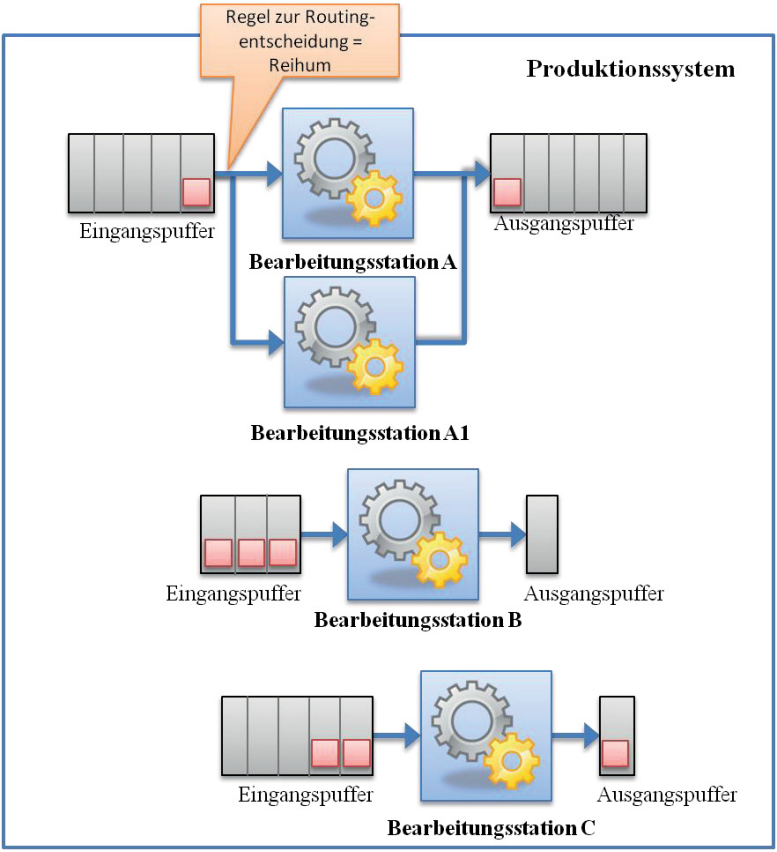


Abbildung 50: Schematische Darstellung der 3x3 Werkstattfertigung mit einer parallelen Bearbeitungsstation

Abbildung in CMSD:

Im Fall alternativ parallel möglicher Bearbeitungsstationen wird das in Anwendungsfall 1 eingeführte Modellierungspattern (vgl. Abbildung 38) entsprechend angepasst, alle parallelen Stationen teilen sich einen einzigen Ein- und Ausgangspuffer (vgl. Abbildung 51). Die Menge der alternativen Stationen kann sowohl ausschließlich gleichartige Entitäten als auch Entitäten, welche in einzelnen oder allen Eigenschaften abweichen, beinhalten. Abweichende Parameter können hierbei alle bisher betrachteten Eigenschaften der Station (RequiredRessource und Ressource) sein, z.B. benötigte Rüstzustände, benötigte Werkerfähigkeiten, Verfügbarkeit usw. In dieser

Interpretation nicht vorgesehen sind Abweichungen arbeitsplanspezifischer Daten (ProcessPlan), so ist u.a. eine unterschiedliche Bearbeitungszeit je alternativer Station nicht realisierbar. Werden alternative Bearbeitungszeiten gewünscht, bieten sich stattdessen alternative Vorgänge (Process) an.

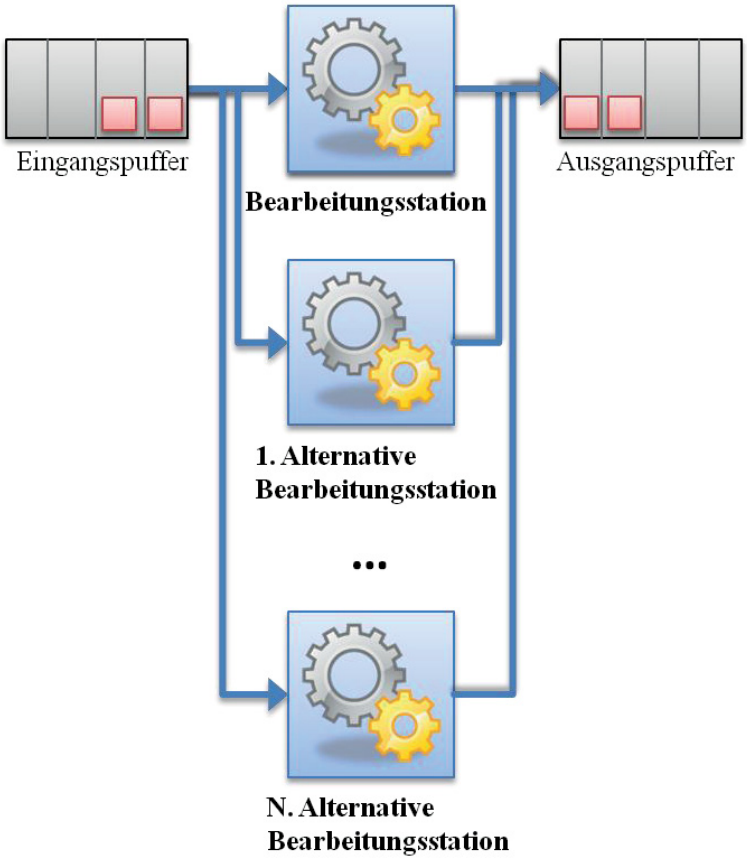


Abbildung 51: Angepasstes Modellierungspattern bei parallelen Stationen

Die Abbildung der zusätzlichen Bearbeitungsstation erfolgt analog allen bisher definierten Stationen, d.h. alle Attribute müssen bzw. können ebenso gesetzt werden (vgl. Tabelle 19, Tabelle 24 und Tabelle 28). Zusätzlich müssen die ResourceGroup Definitionen (vgl. Tabelle 19) aller beteiligten Entitäten (d.h. der Ein- und Ausgangspuffer sowie alle parallelen Stationen) entsprechend gesetzt werden. Hierzu sind vor allem die alternativen Stationen als auch die entsprechenden Verbindungen

von und zu den Puffern aufzunehmen. Um die ohnehin teilweise redundanten Informationen zu minimieren, werden in jeder Entität ausschließlich die verbundenen Entitäten betrachtet, d.h. im Eingangspuffer werden als Gruppenmitglieder neben ihm selbst alle Stationen hinterlegt sowie alle Verbindungen zu diesen (ToRessource) gepflegt, in einer Station hingegen werden nur drei Gruppenmitglieder (Ein- und Ausgangspuffer und die Station selbst) sowie die Verbindungen vom Eingangspuffer (FromRessource) und zum Ausgangspuffer (ToRessource) angelegt (vgl. Abbildung 52).

Zusätzlich benötigt der Eingangspuffer Informationen bzgl. der Routingstrategie (vgl. Tabelle 29), diese wird analog der Reihenfolgestrategie gehandhabt.

Tabelle 29: Erweiterung der CMSD Entität Ressource (Eingangspuffer) zur Realisierung paralleler Bearbeitungsstationen

CMSD Entität	Relevante Attribute (Kardinalität)	Datentyp (Wertebereich)	Beschreibung / Interpretation
Resource	RoutingRule (Property) {nur für Eingangspuffer (1)}	RoutingRule *	Angabe Routingstrategie, (Standard = Reihum/roundRobin)

* zusätzlich entworfener Aufzählungsdatentyp, siehe Anhang C - Die wichtigsten CMSD Aufzählungsdatentypen

Neben den Erweiterungen der statischen Komponenten des Modells, müssen ebenfalls entsprechende Anpassung der Arbeitspläne (ProcessPlan) erfolgen. Hierzu werden in allen Prozessen, die auf mehr als einer Station einer Gruppe bearbeitbar sind, zusätzliche RessourceRequired Blöcke (vgl. Tabelle 20) für jede alternative Station hinzugefügt. Dies ermöglicht nicht nur die Angabe der alternativen Station sondern vor allem auch die Definition der Spezifika des alternativen Bearbeitungsprozesses wie abweichende Rüstzustände oder benötigte Werkerfähigkeiten (vgl. Abbildung 53). Ebenso lassen sich durch dieses Vorgehen für einzelne Arbeitspläne bzw. Produkte ggf. Stationen der Parallelmaschinengruppe ausschließen.

```
...<Resource>
    <Identifier>Ma1_Eingangspuffer</Identifier>
    <Description>Eingangspuffer für Ma1</Description>    ...
    <GroupDefinition>
        <ResourceGroupMember>
            <ResourceIdentifier>Ma1</ResourceIdentifier>
        </ResourceGroupMember>
        <ResourceGroupMember>
            <ResourceIdentifier>Ma1_A</ResourceIdentifier>
        </ResourceGroupMember>
        <ResourceGroupMember>
            <ResourceIdentifier>Ma1_Eingangspuffer</ResourceIdentifier>
        </ResourceGroupMember>
        <Connection>
            <Identifier>Connection_2</Identifier>
            <ToResource>
                <ResourceIdentifier>Ma1</ResourceIdentifier>
            </ToResource>
            <ToResource>
                <ResourceIdentifier>Ma1_A</ResourceIdentifier>
            </ToResource>
        </Connection>
    </GroupDefinition>
    <Property>
        <PropertyDescription>
            <PropertyDescriptionIdentifier>Property_capacity</PropertyDescriptionIdentifier>
        </PropertyDescription>
        <Name>capacity</Name>
        <Description>Aufnahmekapazität des Puffers</Description>
        <Unit>unit</Unit>
        <Value>100</Value>
    </Property>
    <Property>
        <PropertyDescription>
            <PropertyDescriptionIdentifier>Property_decisionRule
            </PropertyDescriptionIdentifier>
        </PropertyDescription>
        <Name>decisionRule</Name>
        <Description>Prioritätsregel des Puffers</Description>
        <Unit>Name</Unit>
        <Value>FIFO</Value>
    </Property>
    <Property>
        <PropertyDescription>
            <PropertyDescriptionIdentifier>Property_decisionRule
            </PropertyDescriptionIdentifier>    </PropertyDescription>
        <Name>routingRule</Name>
        <Description>Routingregel des Puffers</Description>
        <Unit>Name</Unit>
        <Value> roundRobin</Value>
    </Property>
</Resource>...
```

Abbildung 52: Eingangspuffer bei parallelen Bearbeitungsstationen - Auszug einer CMSD Datei

```
<ProcessPlan>
...
    <Process>
        <Identifier>PP1_ProcessA</Identifier>
        <Description>Prozess A des Prozessplans 1</Description>
        <ResourcesRequired>
            <Description>PP1_PA_Ma</Description>
            <MinimumNumber>1</MinimumNumber>
            <MaximumNumber>1</MaximumNumber>
            <Resource>
                <ResourceIdentifier>Ma1</ResourceIdentifier>
            </Resource>
            <AllowableSetup>
                <SetupDefinitionIdentifier>Setup_S1_Ma1</SetupDefinitionIdentifier>
            </AllowableSetup>
            <RequiredEmployeeSkill>
                <SkillDefinitionIdentifier>Skill_A1</SkillDefinitionIdentifier>
                <SkillLevelIdentifier>Skill_A1Level_1</SkillLevelIdentifier>
            </RequiredEmployeeSkill>
        </ResourcesRequired>
        <ResourcesRequired>
            <Description>PP1_PA_Ma_alt</Description>
            <MinimumNumber>1</MinimumNumber>
            <MaximumNumber>1</MaximumNumber>
            <Resource>
                <ResourceIdentifier>Ma1_A</ResourceIdentifier>
            </Resource>
            <AllowableSetup>
                <SetupDefinitionIdentifier>Setup_S1_Ma1_A</SetupDefinitionIdentifier>
            </AllowableSetup>
            <RequiredEmployeeSkill>
                <SkillDefinitionIdentifier>Skill_A1</SkillDefinitionIdentifier>
                <SkillLevelIdentifier>Skill_A1Level_1</SkillLevelIdentifier>
            </RequiredEmployeeSkill>
        </ResourcesRequired>
        <OperationTime>
            <Unit>minute</Unit>
            <Value>30</Value>
        </OperationTime>
    </Process>
..
</ProcessPlan>
```

Abbildung 53: Arbeitsplan für Vorgänge auf parallelen Bearbeitungsstationen - Auszug einer CMSD Datei

Im hier nicht näher betrachteten und eher theoretischen Fall, dass mehr als eine Bearbeitungsstation gleichzeitig von einem Prozess allokiert werden soll, ist eine Abbildung von mehreren RessourceReferences in einem RessourceRequired Block denkbar.

3.3.5 Anwendungsfall 5 - Demontage

Anwendungsfallbeschreibung:

In den bisherigen Anwendungsfällen wurden reine Bearbeitungsprozesse betrachtet, neben diesen sind in vielen Produktionssystemen aber auch Montageprozesse (vgl. Abschnitt 3.3.6) und Demontageprozesse anzutreffen.

Die im Folgenden betrachteten Demontageprozesse[24] sind dadurch charakterisiert, dass als Prozessergebnis mehrere, für die Zielstellung relevante, physische Bauteile oder (Zwischen-) Produkte erzeugt werden. Diese Bauteile können sowohl homogener als auch inhomogener Natur sein und in theoretisch beliebiger Zahl auftreten (vgl. Abbildung 54)

Bei homogenen Prozessergebnissen (Fall A), d.h. das Ergebnis des Prozessschrittes ist in einer definierten Anzahl von gleichen Zwischen- oder Endprodukten zu sehen, fungiert die Demontagestation allein als reiner Multiplikator.

Im Fall inhomogener Prozessergebnissen (Fall B) werden definierte Mengen verschiedener Produktarten im Zuge der Demontage erzeugt. Diese sind im Folgenden aber ggf. unterschiedlich weiter zu bearbeiten. So ist Fall B eine Erweiterung des Fall A um unterschiedliche Arbeitspläne einzelner Zwischenprodukttypen.

Ein zusätzlicher, vor allem in Fall B relevanter Systemparameter betrifft die Anzahl der Ausgangspuffer. So sind sowohl Szenarien denkbar, in denen alle (wenn auch unterschiedliche) Produkte einen Ausgangspuffer durchlaufen oder in denen für jede Produktart ein eigener Ausgangspuffer definiert ist. Diese Unterscheidung hat hierbei aber allein Auswirkungen im Falle einer auftretenden Blockade des Ausgangspuffers. Ist diese weitestgehend auszuschließen, kann der einfachere erste Fall gewählt werden.

Konkret wird das bisher definierte Modell in zwei Schritten erweitert. Es wird für alle Vorgänge, die auf Bearbeitungsstation B bearbeitet werden, angenommen, dass eine Demontage (z.B. in Form eines Zerschneidens) stattfindet. Zunächst wird eine homogene Demontage (Fall A) unterstellt, wobei konstant je Bearbeitungsvorgang drei Produkte erzeugt werden, welche anschließend je einen gegenüber dem Basisszenario unveränderten Bearbeitungsprozess durchlaufen.

[24] "Zerlegen ist ein Fertigungsverfahren nach DIN 8591, in der Praxis auch Demontage [..] genannt. Es beschreibt das Trennen - von Baugruppen, Maschinen oder Anlagen – im Gegensatz zur Montage, als Fertigungsverfahren Fügen. Demontagetechnik ist somit die Umkehrung der Montagetechnik mit dem Ziel der Zerlegung eines komplexen Systems in Subsysteme wie Baugruppen oder einzelne Bauteile" [WIKI2013c].

Beispielsweise erfolgt für alle Aufträge mit dem Endprodukt A die Demontage (Zerlegung) in Prozessschritt 2, nach dieser werden für alle 3 entstanden Zwischenprodukte Bearbeitungen auf Bearbeitungsstation C (Prozessschritt3 des Prozessplanes A) fällig.

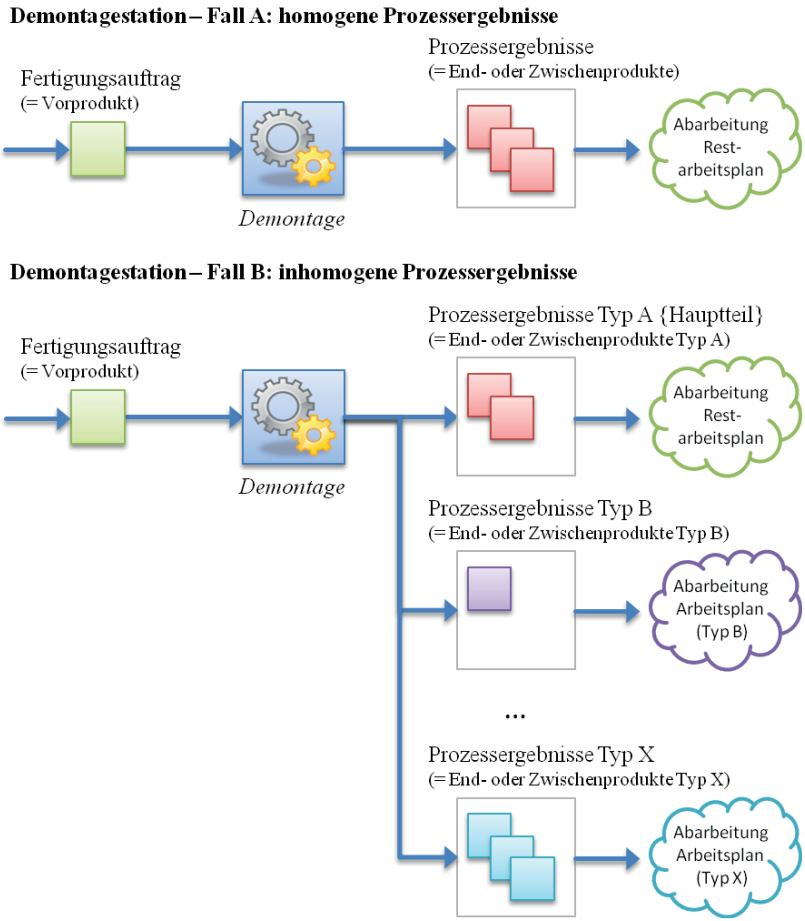


Abbildung 54: Schematische Darstellung der Fälle der Demontage, am Beispiel eines Fertigungsauftrages

Zur Demonstration des Falls B wird der Prozessplan C dahingehend erweitert, dass neben den drei Hauptzwischenprodukten noch zwei Nebenprodukte anfallen. Die Nebenprodukte werden als Produkt C1 vermarktet, bedürfen aber einer weiteren Bearbeitung auf Bearbeitungsstation A wobei angenommen wird, dass dieselben

Prozesseigenschaften (Bearbeitungszeit, benötigter Rüstzustand usw.) wie im ersten Schritt des Prozessplans C (Station B) vorliegen. Auf getrennte Ausgangspuffer wird verzichtet.

Abbildung in CMSD:

Grundlegend lassen sich alle Demontagevorgänge einfach in CMSD definieren. Hierzu ist allein die Nutzung des PartProduced Attributes nötig. Mittels diesem lassen sich sowohl auf Prozessplanebene (vgl. Tabelle 20) als auch auf Vorgangsebene (Process) Mengen sowie Typen von erzeugten Produkten beschreiben (vgl. Tabelle 30).

Tabelle 30: Erweiterung der CMSD ProcessPlan /Process sowie Ressource zur Realisierung von Demontageprozessen - Anwendungsfall 5

CMSD Entität	Relevante Attribute (Kardinalität)			Datentyp (Wertebereich)	Beschreibung / Interpretation
ProcessPlan	Process (1..N)	PartProduced (1..*)	Description (0..1)	String	Kurzbeschreibung des Produkts des Vorgangs
			PartType (1)	PartTypeReference	Referenz auf einen Produkttyp
			PartQuantity (1)	NonNegativeInteger	Anzahl der durch den Vorgang erzeugten Produkte
Ressource	ResourceClass (0..1)			RessourceClass Reference	Referenz auf die Ressourcenklasse (siehe Tabelle 31)

Die Realisierung des Falls A erfolgt allein durch das Setzen der entsprechenden Werte, wobei als PartType einfach weiterhin das Endprodukt nutzbar ist und die PartQuantity den Wert 3 annimmt (vgl. Abbildung 55).

```
...
<Process>
    <Identifier>PP1_ProcessB</Identifier>
    <Description>Prozess B des Prozessplans 1</Description>
    <PartsProduced>
        <Description>DisassamblyPart</Description>
        <PartType>
            <PartTypeIdentifier>PartType1</PartTypeIdentifier>
        </PartType>
        <PartQuantity>3</PartQuantity>
    </PartsProduced>
    <ResourcesRequired>
    ...
</Process>
...
```

Abbildung 55: Prozessplan zur Abbildung von Demontage homogener Bauteile- Auszug einer CMSD Datei

Zusätzlich zu der Definition der Demontage im Processplan empfiehlt sich eine Kennzeichnung der Bearbeitungsstation als Demontagestation. Dies erfolgt hier durch das Nutzen einer einheitlich definierten RessourceClass (vgl. Tabelle 31, Abbildung 56), mit den Parametern: Identifier "Disassembly" und den ResourceType "station". Eine vollständige Liste aller empfohlenen Ressourcenklassen ist dem Anhang zu entnehmen.

Tabelle 31: Zusätzlich empfohlene CMSD Entitäten - Anwendungsfall 5

CMSD Entität	Relevante Attribute (Kardinalität)	Datentyp (Wertebereich)	Beschreibung / Interpretation
ResourceClass	Identifier (1)	Identifier	Eineindeutige ID der Ressourcenklasse, Pflichtfeld nach CMSD Standard
	Description (0..1)	String	Kurzbeschreibung der Ressourcenklasse
	Name (0..1)	String	Optionaler "sprechender" Name der Fähigkeit
	ResourceType (0..1)	resourceType {Station = "machine" "station"}	Typisierung der Ressource

```
...
<ResourceClass>
      <Identifier>Disassembly</Identifier>
      <Description>Ressourcenklasse für Demontagestation</Description>
      <ResourceType>station</ResourceType>
      <Name>Disassembly</Name>
</ResourceClass>
...
<Resource>
      <Identifier>Ma2</Identifier>
      <Description>Bearbeitungstation B</Description>
      <ResourceType>machine</ResourceType>
      <ResourceClass>
            <ResourceClassIdentifier>Disassembly</ResourceClassIdentifier>
      </ResourceClass>
      ...
</Resource>
...
```

Abbildung 56: Kennzeichnung von Demontagestationen in CMSD- Auszug einer CMSD Datei

Im erweiterten Fall B müssen neben der Kennzeichnung der Station vor allem alle erzeugten Produkttypen mit Mengen als PartProduced Attribut des Vorgangs (Process) angelegt werden. Hierzu sind ggf. neuen Produkttypen (vgl. PartTyp Tabelle 20) zu definieren. Die Herausforderung dieses Falls ist aber, dass neben des bestehenden Arbeitsplans (Processplan) der analog dem Fall A weiterhin für das Hauptteil [25]gültig ist weitere (Zwischen-) Produkte auftreten. Für diese muss es möglich sein eigene Vorgangsfolgen in den Arbeitsplan zu integrieren bzw. eigene Arbeitspläne zu definieren bzw. zu referenzieren.

Hierzu sind verschiedenste Abbildungsmöglichkeiten denkbar. So können innerhalb der Prozesspläne alle Vorgänge definiert und entsprechende Vorgänger-Nachfolgerbeziehungen, mittels Prozessbedingungen (ProcessConstaints; vgl. Abbildung 57) oder Subarbeitsplänen (SubProcessGroup), definiert werden, eine weitere Möglichkeit besteht in dem Anlegen von Folge- bzw. Subfertigungsaufträgen (Jobs).

Jede der angerissenen Varianten hat je nach Anwendungskontext Vor- und Nachteile. Vor allem der Aufwand der Modellierung kann je Variante und Kontext schwanken. Im Rahmen dieser Arbeit wird die Abbildung mittels Folgeaufträgen gewählt. Dieser Ansatz lässt sich leicht in CMSD abbilden und weist keine funktionalen Einschränkungen auf. Diesen Vorteilen steht vor allem ein erhöhter Implementierungsaufwand in den Modellgeneratoren gegenüber.

[25] Es wird die Annahme getroffen, dass jeder Vorgang ob Montage oder Demontage einen Haupt und ggf. mehrere Nebenteile verbraucht bzw. erzeugt. Im CMSD Modell sind dies je die als erstes definierten PartConsumed bzw PartProduced Referenzen.

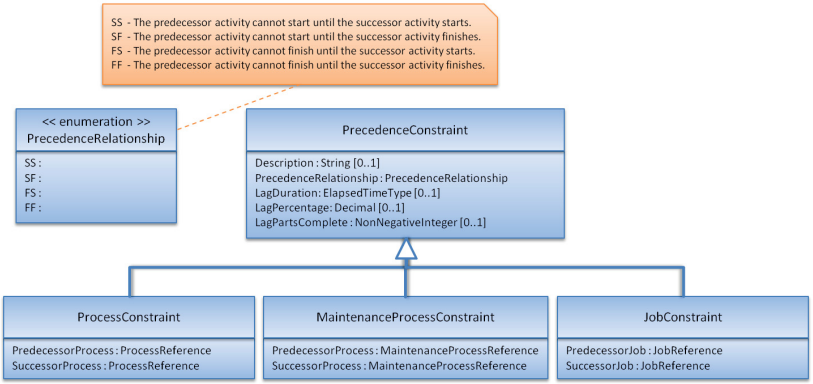


Abbildung 57: CMSD Constraints für Vorgänge und Aufträge (in Anlehnung an [SISO2012, S. 23, 56, 61]

Sollen ggf. mehrere Ausgangspuffer genutzt werden, sind diese entsprechend anzulegen und analog dem bisherigen Vorgehen der Ressourcengruppe hinzuzufügen. Die Zuordnung der Endprodukte zu den einzelnen Ausgangspuffern kann z.B. dynamisch erfolgen, d.h. bei Auftreten verschiedener Zwischenproduktarten werden diese soweit wie möglich auf die Puffer verteilt. Da eine standardkonforme Erweiterung durch Properties für die PartsProduced Attribute nicht vorgesehen ist, ist eine direkte Zuordnung dieser zu einzelnen Ausgangspuffern nur über die unsaubere Umdeutung des Feldes Description zu erreichen (vgl. Tabelle 20).

3.3.6 Anwendungsfall 6 - Montage

Anwendungsfallbeschreibung:

Neben der Demontage ist die Montage[26] ein weiterer typischer Fertigungsprozess. Prinzipiell weist die Montage viele Gemeinsamkeiten mit der Demontage auf. So sind wiederum Fall A die Montage homogener Teile und Fall B die Montage inhomogener Teile zu unterscheiden (vgl. Abbildung 58). Auch können unter Umständen zusätzliche Puffer abzubilden sein, dabei sind Lösungen mit einem Eingangspuffer bis hin zu Lösungen mit Puffern je beteiligtem Teiletyp denkbar.

[26] In der industriellen Fertigung wird unter Montage der geplante Zusammenbau von Bauteilen und/oder Baugruppen zu Erzeugnissen (Produkten) bzw. zu Baugruppen höherer Erzeugnisebene verstanden. Die Umkehrung der Montage ist die Demontage [WIKI2013d].

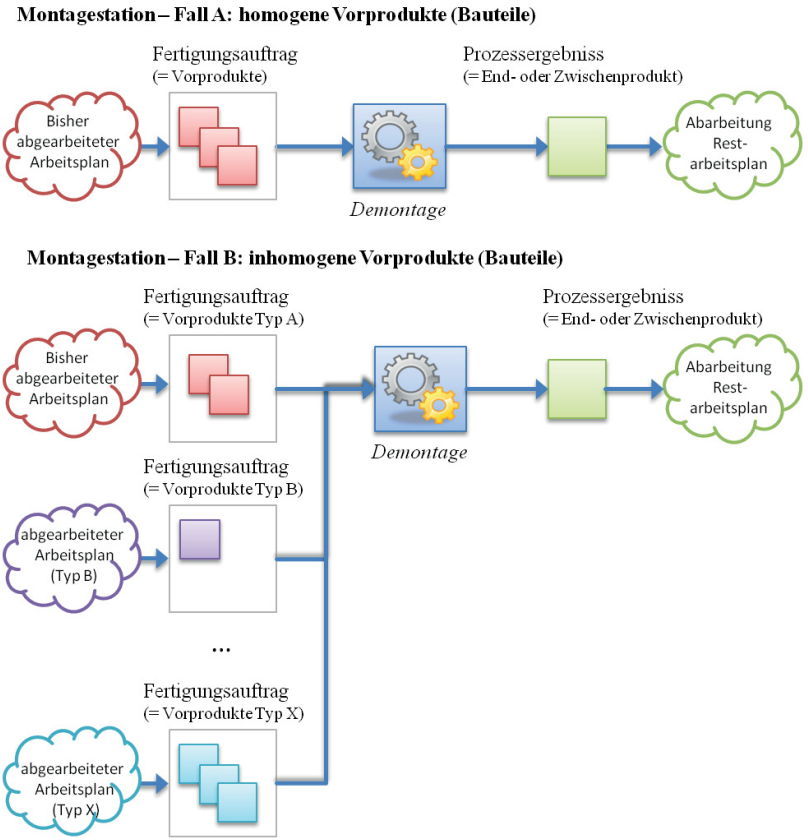


Abbildung 58: Schematische Darstellung der Fälle der Montage, am Beispiel eines Fertigungsauftrages

Konkret wird das bisher definierte Modell in 2 Schritten erweitert, es wird eine zusätzliche Bearbeitungsstation D angelegt, welche grundlegend der Bearbeitungsstation B entspricht, aber statt zur Demontage zur Montage genutzt wird. Des Weiteren wird der Arbeitsplan des Produktes C um eine vierten Vorgang (Process) zur Endmontage erweitert, die Details zum Vorgang können Tabelle 32 entnommen werden.

Tabelle 32: Eigenschaften des Vorgangs 4 zur Montage von Produkt C - Fall A homogene Bauteile

Bearbeitungs-station	Benötigter Rüstzustand	Operationszeit (in Minuten)	Benötigte Werkerfähigkeit	Anzahl von Bauteilen
D	RZ 3	15:00	Skill_A1 (Level 1)	3

Im Fall B werden zusätzlich 2 Produkte B und ein Produkt A während der Montage benötigt.

Abbildung in CMSD:

Die Realisierung des Falls A erfolgt allein durch das Setzen der entsprechenden Werte (PartConsumed - vgl. Tabelle 33), wobei als PartType einfach das Endprodukt nutzbar ist und die PartQuantity den Wert 3 annimmt. Zusätzlich zu der Definition der Montage im Processplan empfiehlt sich eine Kennzeichnung der Bearbeitungsstation als Montagestation. Dies erfolgt analog der Demontagestation durch das Setzen des Attributes der Ressource auf eine entspreche RessourceClass mit den Identifier "Assembly" und dem ResourceType "station".

Tabelle 33: Erweiterung der CMSD ProcessPlan /Process zur Realisierung von Montageprozessen

CMSD Entität	Relevante Attribute (Kardinalität)			Datentyp (Wertebereich)	Beschreibung / Interpretation
ProcessPlan	Process (1..N)	PartConsumed (1..*)	Description (0..1)	String	Kurzbeschreibung des Vorprodukts des Vorgangs
			PartType (1)	PartTypeReference	Referenz auf einen Produkttyp
			PartQuantity (1)	NonNegativeInteger	Anzahl der durch den Vorgangs verbrauchten Vorprodukte

Fall B ist prinzipiell mit verschiedenen Ansätzen abbildbar, so sind neben den hier gewählten separaten Arbeitsplänen zur Zuführung der Bauteile auch analog zur Demontage Arbeitspläne mit mehreren Strängen, unter Nutzung der Vorgänger/Nachfolger Prozessbedingungen (ProcessConstaints; vgl. Abbildung 57), Prozessgruppen (ProcessGroup) oder (interne) Folgeaufträge (Job; vgl. Abbildung 57) denkbar.

Der hier gewählte Ansatz separater Arbeitspläne bedingt wenig zusätzlichen CMSD Interpretationsaufwand bei maximalen Freiheitsgraden der Modellierung. Zunächst sind alle Anpassungen analog zu Fall A zu realisieren, wobei mehrere PartsConsumed Attribute gesetzt werden können, welche die einzelnen Baugruppen beschreiben. Analog zur Demontage und Fall A ist auch hier das erste definierte Bauteil als Hauptteil des Arbeitsplans anzusehen. Für sämtliche weitere Bauteile wird angenommen, dass diese zugeliefert werden, d.h. eigene Aufträge und separate Arbeitspläne modelliert sind. Um Arbeitspläne von Bauteilen der Montage von den Endprodukten zu unterscheiden und die korrekte Bereitstellung an der Montagestation zu gewährleisten, müssen Arbeitspläne der Bauteile im letzten Prozessschritt einen "Lagereintrag" [27] aufweisen. Dieser Lagereintrag beinhaltet keine Bearbeitungsstation, -zeit usw. sondern allein den Puffer (oder das Lager) in welchen das Bauteil abgelegt werden soll. Dieses Vorgehen in Verbindung mit der Definition zusätzlicher Eingangspuffer in der Ressourcengruppe der Montagestation ermöglicht die Realisierung von unterschiedlichen Lagern, z.B. getrennte Lager für jedes Bauteil.

3.3.7 Anwendungsfall 7 - sonstiges

Anwendungsfallbeschreibung:

Neben den bereits beschriebenen typischen Anwendungsfällen kann eine Vielzahl weiterer Fälle auftreten. Die Abhandlung aller Fälle würde nicht nur den Umfang der Arbeit sprengen sondern ist schlichtweg als unmöglich zu betrachten. Im Folgenden ist mit den Sammelprozessen ein spezieller Fall, der sich auch in dieser Form erst im Rahmen der Feldexperimente als relevant zeigte, aufgeführt. Anhand dieses soll die Erweiterbarkeit des Standards dokumentiert werden. Des Weiteren sollen an dieser Stelle Interpretationsansätze weiterer, zum Teil offener Fälle andiskutiert werden.

Im Feldexperiment bei einem lokalen KMU wurde im Rahmen der Modellierung die Funktionalität von Sammelprozessen benötigt, d.h. einige Vorgänge werden nicht je Auftrag oder (Zwischen-) Produkt sondern für eine definierte Menge parallel durchgeführt, Beispiele für solch einen Prozess sind Trocknungsöfen oder Waschprozesse, welche nur bei voll ausgelasteten Stationen gestartet werden. Konkret soll hier eine Station immer genau 20 Aufträge bzw. Produkte parallel bearbeiten, eine Teilbeladung wird ausgeschlossen.

[27] Für Prozesspläne, die keinen solchen Lagereintrag aufweisen, wird angenommen, dass das Endprodukt das System verlässt.

Ebenfalls wurde es im Feldexperiment nötig sogenannte Schiebeöfen abzubilden, diese zeichnen sich dadurch aus, dass sie eine Menge von Aufträgen bzw. Produkten gleichzeitig aufnehmen können, wobei im Gegensatz zum Sammelprozess jedes seine eigene Bearbeitungsstart- und Endzeit aufweist.

Ein weiterer Anwendungsfall betrifft die approximative Abbildung der Förderprozesse, d.h. es sollen nicht Wege, einzelne Stapler oder Förderbänder sondern allein der Zeitverbrauch des Transports abgebildet werden. Konkret soll zwischen Station A und B ein Fördersystem vorhanden sein welches unbeschränkt viele Teile transportieren kann und eine Transportzeit von einer Minute aufweist.

Abbildung in CMSD:

Eine Möglichkeit jederzeit zusätzliches, sehr fallspezifisches Verhalten zu implementieren ohne umfangreiche Uminterpretationen oder zusätzliche Properties zu nutzen ist, wie bereits in der Montage und Demontage angedeutet, die Nutzung der Ressourcenklassen. So kann der Sammelprozess ggf. auch durch zusätzliche Properties abgebildet werden. Hier wurde aber der einfachere Ansatz der Nutzung von Ressourcenklassen gewählt. Im Fall der Sammelprozesse wird jeder Bearbeitungsstation, welche nur Sammelprozesse erlaubt, die Ressourcenklasse " GatheringX ", wobei das X stellvertretend für die Anzahl der gleichzeitige zu bearbeitenden Produkte steht (beispielsweise Gathering20), zugewiesen (vgl. Anhang D). Dieses Vorgehen bedingt zwar erhöhten Aufwand während der Modellerstellung durch den Modellgenerator aber keinerlei Anpassung der weiteren Komponenten bzw. keine zusätzliche Anpassung des Standards.

Auch das zweite Beispiel ist schnell, ohne zusätzliche Attribute mittels der Nutzung einer speziellen Ressourcenklassen, in diesen Fall "Slide_Oven" (vgl. Anhang D) zu realisieren, weitere Anpassung sind nicht nötig.

Für das dritte Beispiel, der Abbildung des Zeitverbrauchs von Förderanlegen, können Dummyvorgänge in den Arbeitsplan sowie entsprechende Dummystationen eingefügt werden. Diese zusätzlichen Stationen und Vorgänge unterscheiden sich prinzipiell nicht von Bearbeitungsstationen und -vorgängen des Anwendungsfalls 1, allein der RessourceType der Station ist auf den Wert "carrier" zu setzen.

3.3.8 Anwendungsfall 8 - Initialisierung

Anwendungsfallbeschreibung:

Wie bereits in Abschnitt 2.1.6 und 2.2.2 diskutiert und im angepassten Vorgehensmodells (vgl. Abschnitt 3.1) zu erkennen, ist die Initialisierung des Simulationsmodells häufig wünschenswert bzw. zwingend nötig. So sind unter anderem betriebsbegleitende Anwendungen wie z.B. simulationsbasierte Absicherung des Produktionsprogramms der nächsten Planungsperiode, meist auf eine hinreichend korrekte Abbildung des aktuellen IST-Zustandes bzw. des zum Start des Betrachtungszeitraums geplanten Zustandes des gesamten Systems angewiesen. Im Zuge der Initialisierung können die aktuellen Zustände aller Entitäten des Systems relevant werden vgl. [BSS2011a, BSS2011b].

Tabelle 34: Im System bereits begonnene Aufträge (Auszug) - Anwendungsfall 8

ID des Fertigungs-auftrags	Produkt	Aktueller Prozess	Status	Rest-bearbeitungs-zeit	Freigabe-termin
Job_PP1_init1	Produkt A	1	In Bearbeitung (Station A)	8:00	15.07.2012 19:00
Job_PP1_init1	Produkt A	1	Wartend (Eingangs-puffer A)	-	15.07.2012 19:30
Job_PP2_init1	Produkt B	2	Rüstend (Station B)	-	15.07.2012 19:00
Job_PP3_init1	Produkt C	2	Unterbrochen /Störung (Station C)	7:45	15.07.2012 14:00

Das bisherige Szenario wird im Folgenden durch Zustandsinformationen der bereits modellierten Entitäten ergänzt. Konkret müssen zu einem definierten Zeitpunkt, hier der 16.07.2012 05 Uhr, die aktuellen Zustände aller Ressourcen abgebildet werden, zum anderen sind die sich zu diesem Zeitpunkt bereits im System befindlichen Aufträge zu betrachten (vgl. Tabelle 34). Wie in den bisherigen Ausführungen sind bei den Ressourcen sowohl Bearbeitungsstationen und Puffer als auch ggf. Werker relevant. Im Fall der Bearbeitungsstationen sind der aktuellen Status (betriebsbereit, gestört, usw.) und der aktuelle Rüstzustand von Interesse (vgl. Tabelle 35).

Tabelle 35: Beispiel für Initialisierungsdaten der Bearbeitungsstationen - Anwendungsfall 8

Bearbeitungsstation	aktueller Status	Aktueller Rüstzustand
A	Betriebsbereit	Abgerüstet
A1	Arbeitend	RZ1
B	Rüstend	Abgerüstet --> RZ1
C	Gestört	RZ2

Neben den Zuständen der Stationen sind ebenfalls die Werker entsprechend zu initialisieren. Hierbei sind Werker vor allem durch ihre aktuelle Tätigkeit gekennzeichnet. Konkret haben von den vorab definierten acht Werkern zum Initialisierungszeitpunkt vier Werker Schichtfrei (E-H), die Tätigkeiten der vier verbleibenden Werker sind Tabelle 36 zu entnehmen.

Tabelle 36: Beispiel für Initialisierungsdaten der Werker - Anwendungsfall 8

Werker	Aktuelle Tätigkeit
A	Bearbeitet Auftrag PP_1_init1 auf Bearbeitungsstation A1
B	Rüstet Bearbeitungsstation B auf RZ1
C	Entstört Bearbeitungsstation C
D	Pausiert

Bei der Initialisierung ist besonders auf die Konsistenz der Daten zu achten, so bedingt ein sich auf Station A in Bearbeitung befindlicher angefangener Auftrag (Restbearbeitungszeit < Bearbeitungszeit), dass die entsprechende Bearbeitungsstation sich in dem Zustand in Bearbeitung (oder gestört) befindet sowie den korrekten Rüstzustand aufweist. Ebenso sind entsprechende Werker zuzuordnen.

Abbildung in CMSD:

Zur Abbildung werden keine zusätzlichen CMSD Entitäten benötigt, wohl aber zusätzliche Attribute inklusive Properties oder Instanzen der bereits genutzten Klassen. Grundlegend ist zunächst der genaue Zeitstempel des Initialisierungszustandes, welcher auch als Startzeit des Simulationslaufs genutzt wird, zu setzen. Dies kann beispielsweise über die Auswertung der Erzeugungszeit (CreationTime) bzw. des letzten Änderungszeitpunktes (UpdateTime) des Dokuments erfolgen (vgl. Tabelle 37).

Die Abbildung des Rüstzustandes und des Zustands der Bearbeitungsstationen erfolgt mittels entsprechender Attribute der CMSD Entitäten Ressource (vgl. Tabelle 37). Bei der Abbildung des Status der Bearbeitungstationen ist eine Liste möglicher

Ausprägungen (z.B. broken , idle, busy, ...) im CMSD Standard definiert, diese sind für die hier angestrebte detaillierte Abbildung im Kontext der Initialisierung nicht ausreichend und sollte zumindest um Einträge für im Rüsten (setup) und pausierend (paused) ergänzt werden.

Werker als zweite große Gruppe der Ressourcen haben abweichende Anforderungen, so ist zwar ebenfalls der aktuelle Status als Attribut abzubilden, die möglichen Werte bzw. deren Bedeutung sind aber gegenüber den Status der Bearbeitungsstationen abweichend, so kann z.B. ein Werker nicht gestört (broken) aber mit der Beseitigung einer Störung (underMaintenance) betraut sein. Die Information, welche Tätigkeit der Werker aktuell durchführt, wird nur mittelbar über den aktuellen Arbeitsort (Station an der der Werker tätig ist; AssociatedResource) des Werkers abgebildet. Die Abbildung des Arbeitsortes erlaubt über die Analyse des Zustands der Bearbeitungsstation sowie der Analyse des ggf. gerade anliegenden Vorgangs, die Ermittlung der Tätigkeit, z.B. ist die Station gestört ist ein aktuell gebundener Werker mit der Reparatur betraut, die benötigte Fähigkeit kann den Daten der Station entnommen werden, hat die Bearbeitungsstation den Status "in Bearbeitung" kann der anliegende Bearbeitungsvorgang durch Analyse aller Jobs ermittelt werden.

Tabelle 37: Erweiterung der CMSD Entitäten im Zuge der Initialisierung

CMSD Entität	Relevante Attribute (Kardinalität)	Datentyp (Wertebereich)	Beschreibung / Interpretation
Resource	CurrentStatus {Bearbeitungsstationen oder Werker} (1)	ResourceStatus* {Default: idle}	Aktueller Status der Bearbeitungsstation / Werkers;
	CurrentSetup {nur Bearbeitungsstationen} (0..1)	SetupDefinition Reference	Aktueller Rüstzustand der Bearbeitungsstation
	AssociatedResource: {nur Werker} (0..1)	Resource Reference	Aktueller Arbeitsplatz (Bearbeitungsstation) des Werkers

Neben den Ressourcen sind bei der Initialisierung gerade die im System befindlichen Aufträge (Jobs; vgl. Tabelle 37; Abbildung 59) und deren aktueller Zustand von essentieller Bedeutung.

Job	Status (1)		JobStatus	Aktueller Status des Auftrags
	Priority (0..1)		String	Priorität des Auftrags; Abbildung der aktuellen Reigenfolgeentscheidung
	ActualEffort (1)	DueDate (0..1)	Timestamp	Liefertermin des Auftrags
		ReleaseDate (1)	Timestamp	Tatsächlicher Freigabezeitpunkt des Auftrags
		StartTime (0..1)	Timestamp	Tatsächlicher Bearbeitungsbeginn des Auftrags
		EndTime (0..1)	Timestamp	Tatsächliches Bearbeitungsende des Auftrags
		CurrentProcessPlanStep (1)	ProcessReference	Referenz auf den aktuellen Vorgang (Process) des Arbeitsplans
		remainingProcessingTime (Property) (0..1)	Duration	Restbearbeitungszeit des Auftrags auf der aktuellen Station
Header Section	CreationTime oder UpdateTime		Timestamp	Erzeugungs- / Änderungszeit der CMSD Datei, entspricht dem Initialisierungszeitpunkt

* nicht Standardkonform, als Erweiterung des Standards durch den Autor empfohlen, siehe Anhang C - Die wichtigsten CMSD Aufzählungsdatentypen

Arbeitsaufträge (Jobs) sind zunächst hauptsächlich gekennzeichnet durch ihren aktuellen Bearbeitungsstand (CurrentProcessPlanStep), d.h. welcher Bearbeitungsvorgang derzeit durchgeführt wird bzw. im Fall der Zwischenlagerung als nächstes durchzuführen ist. Durch Auswertung des Status des Auftrags (z.B. wartend, in Bearbeitung usw.) und des aktuell anstehenden Bearbeitungsvorgangs in Verbindung mit den hinterlegten Daten der Arbeitspläne, Ressourcen usw. kann der Gesamtzustand recht genau beschrieben werden. So ergibt sich der aktuelle Pufferbestand indem alle

nicht in Bearbeitung befindlichen Aufträge (Status: "blocked") im Eingangspuffer der Bearbeitungsstation liegen, welche im anstehenden Bearbeitungsvorgang definiert werden. Begonnene Aufträge (Status: "started") sind direkt auf der referenzierten Bearbeitungsstation zu verorten. Durch das Auswerten des Stationsstatus kann ermittelt werden ob aktuell die eigentliche Bearbeitung, das Rüsten für die Bearbeitung oder eine Störung vorliegt. Nicht begonnene Aufträge, die auf Abarbeitung warten, werden durch den Status "released" gekennzeichnet, analog weisen beendete Aufträge den Status "completed" auf.

```
...
<Job>
      <Identifier>Job_impA_init1</Identifier>
      <Description>TestJob importiert 61</Description>
      <Status>started</Status>
      <UpdateTime>2012-07-16T15:35:46</UpdateTime>
      <Priority>1</Priority>
      <PlannedEffort>
            <UpdateTime>2012-07-16T15:35:46</UpdateTime>
            <DueDate>2012-07-16T20:00:00</DueDate>
            <ReleaseDate>2012-07-13T07:00:00</ReleaseDate>
...
      </PlannedEffort>
      <ActualEffort>
            <UpdateTime>2012-07-16T15:35:46</UpdateTime>
            <DueDate>2012-07-16T20:00:00</DueDate>
            <ReleaseDate>2012-07-13T07:00:00</ReleaseDate>
            <ProcessPlan>
                  <ProcessPlanIdentifier>ProcessPlan1</ProcessPlanIdentifier>
            </ProcessPlan>
            <CurrentProcessPlanStep>
                  <ProcessPlanIdentifier>ProcessPlan1</ProcessPlanIdentifier>
                  <ProcessIdentifier>PP1_ProcessB</ProcessIdentifier>
            </CurrentProcessPlanStep>
      </ActualEffort>
      <Property>
            <Name>remainingProcessingTime </Name>
            <Unit>minutes</Unit>
            <Value>20.0</Value>
      </Property>
</Job>
...
```

Abbildung 59: Initialisierung eines Auftrags - Auszug einer CMSD Datei

Wie in Abschnitt 2.2.2 diskutiert, kann je nach Szenario eine Verfeinerung der Abbildung hinsichtlich abgearbeiteter Vorgänge sinnvoll sein. Dies kann durch die Nutzung einer zusätzlichen Property zur Abbildung der Restbearbeitungszeit (remainingProcessingTime) realisiert werden, ist diese nicht gesetzt, wird die im Process definierte Bearbeitungszeit angenommen. Alternativ kann der gespeicherte Beginn der

Bearbeitung herangezogen werden. Für Details der Speicherung von Ereignissen sei auf Abschnitt 3.3.9 sowie die theoretischen Ausführungen in Abschnitt 2.2.2 verwiesen.

Besonders bei der Initialisierung sind verschiedene Interpretationen bestehender Attribute bzw. Abbildungsmöglichkeiten denkbar. In dieser Interpretation wird auf das mittelbare Abbilden von Sachverhalten durch das Zusammenspiel der Entitäten und deren Attribute gebaut, z.B. die Ermittlung des aktuellen Bestandes eines Puffers oder die Tätigkeit eines Werkers. Diese Informationen könnten alternativ auch direkt abgebildet werden. Eine direkte Abbildung wäre zwar leichter zu interpretieren, wobei aber zahlreiche Redundanzen entstehen und die Konsistenzsicherung aufwendiger würde.

Die hier aufgeführten initialisierungsrelevanten Attribute sind ausreichend für die im Vorhinein beschriebenen Anwendungsfälle. Im Rahmen möglicher Erweiterungen z.B. der Abbildung von Fördersystemen, Plänen, Laufwegen von Werkern oder einer Betrachtung auf Bauteilebene werden entsprechend zusätzliche Daten benötigt (vgl. [BSS2011a, S. 2234]).

3.3.9 Anwendungsfall 9 - Abbildung der Zustands- und Betriebsdaten

Anwendungsfallbeschreibung:

Alle bisherigen Anwendungsfälle fokussierten auf die Modellierung der Eingangsdaten zur Simulationsmodellerstellung. Diese sind, wie bereits in Abschnitt 2.1.7 diskutiert, durch Daten bzgl. Ergebnisse einzelner Simulationsläufe zu erweitern, dazu sind geeignete Maßnahmen in das CMSD Datenmodell zu integrieren (vgl. [BSS2012]). Um auch anwendungsfallspezifische Auswertungsmöglichkeiten zu ermöglichen, werden nicht einzelne Kennziffern erfasst, sondern Voraussetzungen geschaffen Postsimulationsauswertungen durchzuführen, d.h. es werden allein Grunddaten vergleichbar den BDE-Daten gespeichert [BSS2012, S.4-6]. Neben den sich eröffnenden Ausweitungsmöglichkeiten könne diese Daten, im Zuge der Weiternutzung des Modells, im Rahmen der Initialisierung folgender Simulationsläufe genutzt werden, hierzu sind entsprechend die Zustandsdaten aller Systemobjekte zu speichern (vgl. 3.3.8).

Die im Kontext dieser Arbeit relevanten BDE-Datensätze weisen eine einheitliche Grundstruktur auf, die eine Weiterverarbeitung ermöglicht. Jeder der Datensätze besitzt einen eindeutigen Zeitstempel, eine Ereignistypkennung sowie Referenzen auf alle relevanten beteiligten Entitäten. Praktisch sind drei Gruppen potentiell beteiligter Entitäten identifizierbar. Zunächst ist jedem Datensatz zwingend ein Objekt (Ort des Ereignisses) zuzuordnen, diese Objekte können sowohl Bearbeitungsstationen als auch Lager, Quellen oder Senken sein. Optional je nach Ereignistyp umfassen die Datensätze

Informationen zu beteiligten Aufträgen und deren Bearbeitungsstands (z.B. Angabe des Vorgangs) sowie zu ggf. allokierten Werkern.

Konkret sollen Datensätze mit allen relevanten Informationen angelegt werden wenn:

a. Aufträge freigegeben werden,
b. Aufträge fertiggestellt sind,
c. Bearbeitungsvorgänge beginnen,
d. Bearbeitungsvorgänge enden,
e. Rüstvorgänge beginnen,
f. Störungen auftreten und
g. Störungen behoben wurden.
h. (Start Transportvorgang)
i. (Ende Transportvorgang)[28]

Ereignisse bzgl. Schicht bzw. Pausenbeginn sind ggf. weitere Ereignistypen, welche in den hier betrachteten Szenarien nicht relevant sind, da diese Inforationen deterministisch sind und somit aus den definierten Kalenderdaten (siehe Tabelle 19) direkt auslesbar sind.

Abbildung in CMSD:

Für die Abbildung der Ereignis / BDE-Daten ist die Event Klasse (Abbildung 60) prädestiniert, welche im Support Packet des CMSD Standard definiert ist. Im CMSD Standard sind Ereignisse (Events) nur für Aufträge (JobEffortDescription) vorgesehen, die Limitierung dieser auftragsorientierten Sichtweise ist entsprechend zu beachten. So sind Ereignisse, z.B. Störungen, die keinen Auftrag zuzuordnen sind, nicht direkt abbildbar. Können solche Ereignisse in einem System auftreten und sind diese nicht vernachlässigbar, sind verschiedene Ansätze zur Abbildung denkbar.

Der erste Ansatz ist im Wechsel zu einer ressourcenorientierten Ereignisspeicherung zu sehen, hierzu ist es nötig, Ereignisse (Event) als Eigenschaft der Ressourcen, z.B. Bearbeitungsstation zu behandeln. Dieser Ansatz würde eine vollständige Abbildung aller Ereignisse ermöglichen, zudem herrscht in realen BDE-Datenerfassungsimplementierungen meist ein ressourcenorientierter Ansatz vor.

Als problematisch erweist sich die standardkonforme Abbildung in CMSD, da:

a. in der Ressourcenklasse keine Events als Attribut vorgesehen bzw. erlaubt sind,
b. im Standard keine EventReference Klasse definiert ist und somit keine ReferenceProperty auf Events direkt möglich ist [SISO2012a, S.54].

[28] nur in Verbindung mit den in Anwendungsfall 7 beschrieben Transportstationen.

Somit ist eine standardkonforme Abbildung nur über den Umweg der Definition einer JobReference als Property der Ressourcen denkbar. Der so referenzierte Auftrag kann entsprechend innerhalb der JobEffortDesciption wiederum beliebig viele Ereignisse enthalten. Diese Lösung erscheint aber aufwendig und wenig praktikabel.

Ebenfalls nicht praktikabel erscheint der hybride Ansatz, d.h. die Nutzung des ressourcenorientierten Ansatzes für Ereignisse, die nicht direkt Aufträgen zuordenbar sind. Dieser Ansatz bietet keine zusätzlichen Informationen, erhöht aber den Aufwand analog dem rein ressourcenorientierten Ansatz.

In der hier vorgeschlagenen Interpretation wird daher ein "Dummy"-Auftrag (Name: event_not_assigned) eingeführt, welcher zur Kapselung aller Ereignisse dient, die nicht "echten" Aufträgen zuordenbar sind. Dieses Vorgehen erlaubt eine Abbildung aller Ereignisse ohne Standardverletzungen und ohne weitere extensive Propertienutzung.

Aus Sicht des Autors ist im Rahmen der zukünftigen Standardentwicklung die Ermöglichung von ressourcenorientierten Ereignissen vorzusehen. Dies kann zum einen durch das Erweitern der Event Klasse zur UniqueEntity in Verbindung mit der dadurch möglichen Definition einer EventReference Klasse erfolgen. Hierdurch wäre es möglich Events als Properties z.B. an Ressourcen zu nutzen. Alternativ ist auch die direkte Definition eines zusätzlichen Attributes in der Ressourcenklasse denkbar.

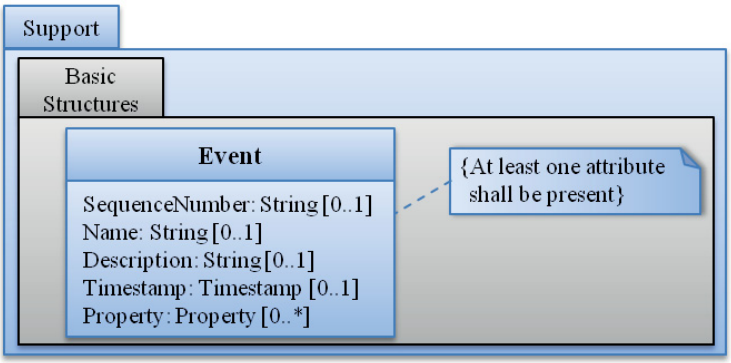


Abbildung 60: Die CMSD Event Klasse [SISO2012a, S.23]

Die Event Klasse (Abbildung 60; Abbildung 61) selbst erlaubt neben der Angabe einer Sequenznummer, welche im folgenden als fortlaufender ganzzahliger Wert definiert wird und für jeden BDE-Datensatz eindeutig ist, zunächst die Angabe des Zeitstempels (timestamp). Die Ereignistypen werden im Namensfeld (Name) hinterlegt, mögliche Ausprägungen sind Anhang C zu entnehmen.

```
...
<Job>
      <Identifier>Job01</Identifier>
      <Status>started</Status>
      <PlannedEffort> …
...
      </PlannedEffort>
      <ActualEffort> …
            <Event>
                  <SequenceNumber>4</SequenceNumber>
                  <Name>start setup</Name>
                  <Timestamp>2012-02-26T09:07:23</Timestamp>
                  <Property>
                        <Name>ProcessStep</Name>
                        <ProcessReference>
                              <ProcessIdentifier>PP1Step1</ProcessIdentifier >
                        </ProcessReference>
                  </Property>
                  <Property>
                        <Name>usedResource</Name>
                        <ResourceReference>
                                    <ResourceIdentifier>Ma3</ResourceIdentifier>
                        </ResourceReference>
                  </Property>
                  <Property>
                        <Name>usedResource</Name>
                        <ResourceReference>
                              <ResourceIdentifier>worker1</ResourceIdentifier>
                        </ResourceReference>
                  </Property>
            </Event>
            <Event>
                  <SequenceNumber>5</SequenceNumber>
                  <Name>start work</Name>
                  <Timestamp>2012-02-26T09:17:32</Timestamp>
                  <Property>
                        <PropertyDescription> … </PropertyDescription>
                        <Name>ProcessStep</Name>
                        <ProcessReference>
                              <ProcessIdentifier>PP1Step1</ProcessIdentifier >
                        </ProcessReference>
                  </Property>
                  <Property>
                        <Name>usedResource</Name>
                        <ResourceReference>
                              <ResourceIdentifier>Ma3</ResourceIdentifier>
                        </ResourceReference>
                  </Property>
            </Event>
      </ActualEffort>
</Job> …
```

Abbildung 61: Ergebnisdatendarstellung am Beispiel des Auftrags 1 - Auszug einer CMSD Datei

Die Zuordnung der Ereignisse zu Aufträgen ist per se gegeben, da alle Events allein als Attribute einer JobEffortDescription auftreten können. Die Zuordnung eines Objektes, welches wie bereits geschildert für alle Ereignisse als obligatorisch anzusehen ist, wird mittels einer RessorceReference-Property realisiert. Optional sind analog RessorceReferencen auf beliebig viele Werker erlaubt (vgl. Abbildung 61).

Neben den BDE-Daten sind der Vollständigkeit halber und um weitere Simulationsläufe zu unterstützen ebenfalls alle Zustands- und Statusinformationen der Bearbeitungsstationen, Werker und Aufträge entsprechend den aktuellen Werten anzupassen (vgl. 3.3.8, Tabelle 37).

3.3.10 Anwendungsfall 10 - Adaption

Anwendungsfallbeschreibung / Abbildung in CMSD:

Ergänzend zu den bisher aufgeführten Anwendungsfällen muss, wie in Abschnitt 2.2.3 diskutiert, die Möglichkeit bestehen Modelle zu adaptieren, d.h. bestehende Modelle anzupassen. Neben der bereits thematisierten Initialisierung als Form der Adaption im weitesten Sinn (Adaption des Zustands) sind auch Änderungen bzgl. grundlegenderer Strukturen, d.h. den technischen und organisatorischen Daten (vgl. Abbildung 5), z.B. der Art und Menge von Bearbeitungsstationen zu ermöglichen. Sicherzustellen ist, dass jegliche Adaption tatsächlich nur die gewollten Änderungen und keine unerwünschten Nebeneffekte bewirkt, ebenso ist sicherzustellen, dass keine inkonsistenten Datenbestände entstehen. Zugleich ist eine Versionierung bzw. eine Historie der Änderungen sinnvoll. Dies kann im einfachsten Fall durch einen Verweis des adaptierten Modells auf das ursprüngliche Modell erfolgen, die so realisierte Historie erlaubt durch Vergleich zweier Modelle die Identifikation des Deltas. Zur Abbildung der Historie auf CMSD Dokumentebene sind einige wenige Attribute zu pflegen, siehe Tabelle 38 und Abbildung 62.

Tabelle 38: Attribute zur Bildung von Historien von CMSD Modellen

CMSD Section	Relevante Attribute (Kardinalität)	Datentyp (Wertebereich)	Beschreibung / Interpretation
Header Section	Version (1)	String (nur Ziffern und Punkte)	Angabe einer Versionsnummer
	CMSDDocumentReference (1..N)	CMSDDocument Reference	Referenzen auf zumindest das direkte Basisdokument; ggf. Referenz auf alle Vorgängerdokumente

In Abschnitt 2.2.3 wurden in diesem Sinn drei Adaptionsoperationen identifiziert, Einfügeoperationen, Änderungsoperationen und Löschoperationen. Viele dieser Operationen wurden ohne von Adaption zu sprechen bereits angewendet. So wurden Einfügeoperationen implizit in einigen Anwendungsfällen genutzt, beispielsweise wurden komplette Entitäten, wie Werker (vgl. 3.3.3) und zusätzliche (parallele) Bearbeitungsstationen (vgl. 3.3.4) hinzugefügt ebenso sind einzelne zusätzliche Attribute aufgenommen worden, z.B. die Reparaturskills von Stationen (vgl. 3.3.3) oder Routingregeln von Eingangspuffern (vgl. 3.3.4). Auch Änderungsoperationen sind bereits Bestandteil bestehender Anwendungsfälle, so wurden verschiedene Attribute, z.B. der Status der Bearbeitungsstation(vgl. 3.3.9) geändert. Löschoperationen wurden bisher nicht angestoßen, können aber als Umkehrung der Einfügeoperationen betrachtet werden und sind für alle Entitäten denkbar, so können z.B. Werker ausscheiden, Bearbeitungsstationen aus dem System entfernt werden oder aber auch nur nicht mehr benötigte Rüstzustände oder Arbeitspläne gelöscht werden.

```
...
<HeaderSection>
    <DocumentIdentifier>adaptedCMSD</DocumentIdentifier>
    <Version>1.1</Version>
    <CreationTime>2013-03-28T14:04:29</CreationTime>
    <Metadata>    </Metadata>
    <CMSDDocumentReference>
        <LocalDocumentIdentifier>basisCMSD</LocalDocumentIdentifier>
        <DocumentLocation>
            https://github.com/SoerenBergmann/CMSD-Interpretation/Szenario3x3_einfach.xml
        </DocumentLocation>
    </CMSDDocumentReference>
    <PropertyDescription>
        <Identifier>Property_ oldIdentifier </Identifier>
        <Name> oldIdentifier </Name>
        <Description>originale Identifier nach einer Änderung </Description>
        <ParentEntityName>IdentifiableEntity</ParentEntityName>
        <PropertyCardinality>
            <Minimum>0</Minimum>
            <Maximum>*</Maximum>
        </PropertyCardinality>
        <PropertyDataTypeDescription>
            <PropertyDataType>simple</PropertyDataType>
            <SimpleValueDataType>String</SimpleValueDataType>
        </PropertyDataTypeDescription>
    </PropertyDescription>
</HeaderSection>
...
```

Abbildung 62: Historienbildung im Zuge der Adaption - Auszug einer CMSD Datei

Betreffen die Löschoperationen vollständige Entitäten bzw. werden Identifier geändert, sind alle vorkommenden Abhängigkeiten zu betrachten, d.h. es ist sicherzustellen, dass keine Referenzen [29] auf diese Entität in anderen Entitäten gesetzt ist. Diese Abhängigkeiten können meist nicht automatisch sondern allein in Interaktion mit dem Modellierer gelöst werden. Hierzu stehen durchaus verschiedene Möglichkeiten zur Verfügung, welche aber immer eine Adaption weiterer Entitäten bedingen. Ein Beispiel für solch einen Fall ist das Löschen eines Rüstzustandes, welcher in Vorgängen eines oder mehrerer Arbeitspläne als benötigt gekennzeichnet ist. In diesem Beispiel sind verschiedenste Lösungsoptionen denkbar, sie reichen vom Verhindern der Löschoperation, über das Ändern des im Arbeitsplan benötigten Rüstzustands auf einen verfügbaren bis hin zum Löschen des betroffenen Arbeitsplans.

Im Fall der Änderungen von Identifiern müssen, um später korrekte Auswertungen und die fehlerfreie Modelladaption in den entsprechenden Simulatoren zu ermöglichen, neben den neuen Identifiern zumindest auch der letzte gültige Identifier gespeichert werden. Hierzu wird eine einfache Text-Property mit dem Name "oldIdentifier" genutzt, welche für alle relevanten Entitäten erlaubt ist (vgl. Abbildung 62).

Festzuhalten ist, dass alle Operationen auf der Ebene des CMSD Konzeptmodells weitestgehend problemlos ohne zusätzliche Anpassungsbedarfe zu realisieren sind, allein das Referenzieren des Basismodells und im Fall der Änderung von Identifier Attributen sind wie beschrieben zusätzliche Daten nötig. Eine nicht zu unterschätzende Aufgabe ist aber die Wahrung der Konsistenz, wie ggf. auch die Vermeidung ungenutzter bzw. nicht mehr genutzter Daten, z.B. sind ggf. nicht mehr referenzierte Rüstzustände usw. entfernbar. Aus Gründen der hohen Komplexität sind hierfür in der softwaretechnischen Implementierung des Frameworks geeignete Maßnahmen zu treffen.

[29] Referenzen in CMSD werden prinzipiell über den Identifier, den jede (Limited) Unique Entity aufweist, realisiert [SISO2012a, S. 67ff]

3.3.11 Fazit

Im Rahmen dieser Arbeit konnte insbesondere im Abschnitt 3.3 Interpretation des CMSD Standards, anhand von zehn typischen Anwendungsfällen der Werkstattfertigung gezeigt werden, dass der CMSD Standard sehr gut als formales Konzeptmodell im Kontext der Simulation von Produktionssystemen geeignet ist. Grundlegende Objekte wie die Ressourcen, d.h. Bearbeitungsstationen und Werker, Rüstmatrizen, Arbeitspläne und Aufträge sind adäquat abbildbar. Ebenso sind komplexere Sachverhalte wie z.B. die Demontage/Montage oder eine sehr detaillierte Zuordnung von Werken zu einzelnen Aufgaben (z.B. Reparaturen, Rüsten) nahtlos in CMSD integrierbar. Schließlich konnte gezeigt bzw. nachgewiesen werden, dass in CMSD ebenfalls Systemzustände, die z.B. zur Initialisierung des Modells genutzt werden können, vollständig abbildbar sind. Die Abbildung von Systemzuständen und die größtenteils problemlose Speicherung von Ereignisinformationen (Events) erlauben CMSD auch zum Austausch von Ergebnissen einzelner Simulationsläufe zu nutzen.

In allen Fällen hat sich gezeigt, dass zur Abbildung im Detail verschiedene Möglichkeiten bestehen, dies reicht von alternativ nutzbaren Attributen (z.B. Jobs: StartTime vs. ReleaseDate) bis hin zu verschiedenen Modellierungsansätzen (z.B. Modellierungspattern, Montage). Die hier vorgeschlagene Interpretation stellt eine in sich konsistente, erweiterbare und im Rahmen der Ziele vollständige Interpretationsvariante dar. Mangels weiterer publizierter Varianten kann kein Vergleich gezogen werden.

In der gesamten Interpretation wurde auf Standardkonformität geachtet, des Weiteren wurden möglichst wenige zusätzliche Properties genutzt (vgl. Anhang A - Erweiterungen des CMSD-Standards / Liste der Properties) sowie nur in zwingenden Fällen bestehende Aufzählungsdatentypen (Enumerations) erweitert (vgl. Anhang C - Die wichtigsten CMSD Aufzählungsdatentypen).

Die Mehrzahl der zusätzlichen Properties sowie die zusätzlichen Werte innerhalb von Aufzählungsdatentypen sind von so essentieller Natur, dass für zukünftige Versionen des Standards vom Autor ausdrücklich empfohlen wird, diese direkte aufzunehmen, d.h. Aufzählungsdatentypen zu erweitern und den Properties entsprechende Attribute in den Entitäten zu definieren. Ebenfalls sinnvoll erscheint eine verbindliche Definition von Verteilungsfunktionen (vgl. 3.3.2, Tabelle 23). Des Weiteren ist im Zuge der Interpretation in einigen wenigen Fällen die hier dokumentierte Interpretation Limitierungen des Standards geschuldet. So sind einige wenige Elemente nicht durch Properties zu erweitern (z.B. PartsProduced vgl. 3.3.6) bzw. es sind keine Referenzproperties auf einzelne Elemente möglich (z.B. Events vgl. 3.3.9). Weitere kleine Limitierungen wurden in den entsprechenden Abschnitten thematisiert.

3.4 Teilkomponenten des Frameworks

In den folgenden Abschnitten werden die prototypisch implementierten Komponenten eines Frameworks zur automatischen Modellgenerierung, -initialisierung, -validierung und -adaption vorgestellt. Diese nutzen konsequent die im Zuge der letzten Abschnitte beschriebenen Konzepte. So basieren alle Komponenten auf der hier dokumentierten CMSD Interpretation (vgl. Abschnitt 3.3) und sind zur Anwendung im Rahmen des erweiterten Vorgehensmodells (vgl. Abschnitt 3.1) entworfen worden.

Die Komponenten ergänzen sich hierbei so, dass eine vollständige Unterstützung einer Simulationsstudie von der Erstellung des Konzeptmodells in CMSD bis hin zur Auswertung abgedeckt ist. Im Folgenden wird die Grobarchitektur des Frameworks, welches aus diversen Komponenten besteht, beschrieben. Des Weiteren wird deren Zusammenspiel bezogen auf eine einzelne Simulationsstudie sowie im Kontext des Produkt- bzw. Produktionslebenszyklus des abgebildeten Systems thematisiert.

3.4.1 Grobarchitektur und Systemübersicht

In Abbildung 63 sind Anforderungen an das Framework über die Planungs-, Inbetriebnahme- bis hin zur Betriebsphase abgebildet. Fokus dieser Arbeit ist aber vor allem die betriebsbegleitende Simulation und die Planungsunterstützung. Es wird, wie schon im Vorgehensmodell diskutiert, deutlich, dass die Nutzung der Simulation aber prinzipiell in allen Phasen möglich ist, sowie dass auch innerhalb einer Phase ggf. mehrmals Simulationsbedarf besteht. So können beispielsweise Meilensteine der Planung oder Änderungen am Produktionssystem während des Betriebs z.B. durch die Realisierung von Lerneffekten, Änderungen aus kontinuierlichen Verbesserungsprozessen usw. Trigger für eine erneute Modellgenerierung oder Adaption des bestehenden Modells sein. Somit können Bedarfe zur Modellgenerierung oder zur Adaption bestehender Modelle durch externe Ereignisse ausgelöst werden, z.B. Erreichen von Meilensteinen oder durch einen Schwellwert überschreitende Abweichungen des Verhaltens des Modells vom Realsystem, welche im Zuge der periodisch oder permanent laufenden (automatischen) Validierung der Simulationsergebnisse erkannt wurde. Des Weiteren ist neben der Modellgenerierung und -adaption in der Grobübersicht der Anforderungen der Bedarf an Schnittstellen zur Erzeugung von CMSD Daten und an geeigneten Werkzeugen zur Bearbeitung dieser, z.B. zur Szenariobildung sowie zur Experimentsteuerung (vgl. CMSD Webfrontend - Abschnitt 3.4.3) und Ergebnisauswertung (vgl. WebStatMonitor - Abschnitt 3.4.4) zu erkennen. Eine bereits in Abschnitt 3.2 thematisierte Sonderrolle spielt die Abbildung des dynamischen Verhaltens.

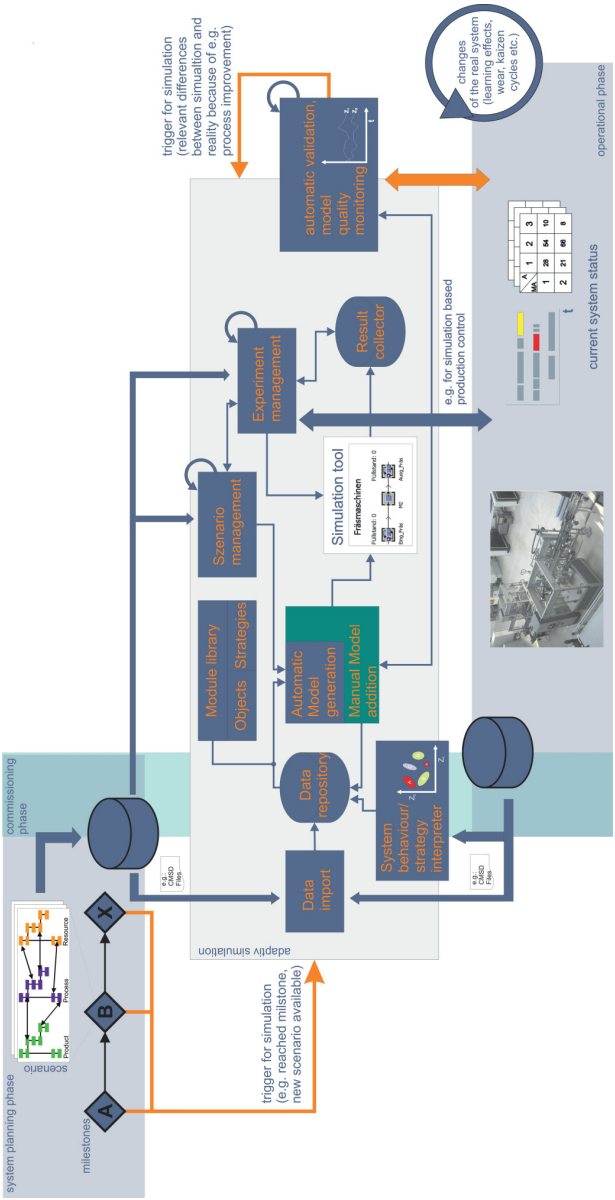


Abbildung 63: Grobübersicht zu Anforderungen des Frameworks zur automatischen Modellgenerierung, -adaption und -validierung [Be2010]

Die Datenquellen können je nach Phase unterschiedlich sein, müssen aber entsprechende Funktionalitäten zum CMSD basierten Datenaustausch bieten. In Abbildung 64 sind zwei mögliche CMSD Datenquellen (ERP und MES) beispielhaft angegeben, analog sind weitere Quellen z.B. aus dem Kontext der Prozessplanung, Konstruktion usw. denkbar.

Neben den Schnittstellen zu z.B. SAP ERP (vgl. [Wü2013]) ist unabhängig einer aktuell durchzuführenden konkreten Studie die Validierung (vgl. Abschnitt 3.4.4) zu nennen. Wie bereits im Zuge des Vorgehensmodells (vgl. Abschnitt 3.1) dargelegt, ist eine Validierung der Simulationsmodelle unverzichtbar. Validierung kann und sollte aber nicht nur einmalig nach der Modellgenerierung durchgeführt werden, sondern vor allem in der betriebsbegleitenden Anwendung periodisch wiederholt werden. Diese Wiederholungen helfen Modelle auch langfristig aktuell zu halten und dienen wie angesprochen ggf. als Trigger für Simulationsmodelladaptionen.

In Abbildung 64 sind die im Zuge eines Simulationslaufs benötigten Teilkomponenten des Frameworks bzw. der Ablauf einer Simulationsstudie im Framework ersichtlich. Die benötigen Komponenten sind die CMSD basierten Modellgeneratoren und Initialisatoren (vgl. Abschnitt 3.4.2), CMSD-Schnittstellen zu Quellsystemen, die Nutzerschnittstelle um CMSD Daten zu pflegen, zu validieren und ggf. anzureichern, Alternativen und Experimente zu definieren (CMSD Webfrontend, vgl. Abschnitt 3.4.3) sowie um Ergebnisse von CMSD basierten Experimenten auszuwerten (WebStatMonitor, vgl. Abschnitt 3.4.4). Die Komponenten sind hierbei, bis auf die Schnittstellen, anwendungsfallneutral, d.h. unabhängig des Anwendungsfalls und der Zielstellung der Simulation nutzbar. Zu bemerken ist, dass eine Simulationsstudie allein unter Nutzung der aufgeführten Komponenten (ggf. zuzüglich der Validierung) durchgeführt werden kann. Hierbei ist das gesamte Framework so gestaltet, dass für den Endnutzer weite Teile transparent sind, so sind die eigentlichen Modellgeneratoren und Simulationsmodelle bzw. Simulatoren für den Endnutzer nicht direkt sichtbar. Eine Erweiterung oder Analyse der Modelle durch Simulationsexperten ist aber, da diese entsprechend gespeichert werden, jederzeit möglich. Die für den Endnutzer entworfenen Komponenten sind alle als reine Webanwendung implementiert, dies ermöglicht die Realisierung der sogenannten webbasierten Simulation bzw. von Simulation as a Service (vgl. [Be2012]). Alle Komponenten werden im Folgenden separat ausführlich thematisiert.

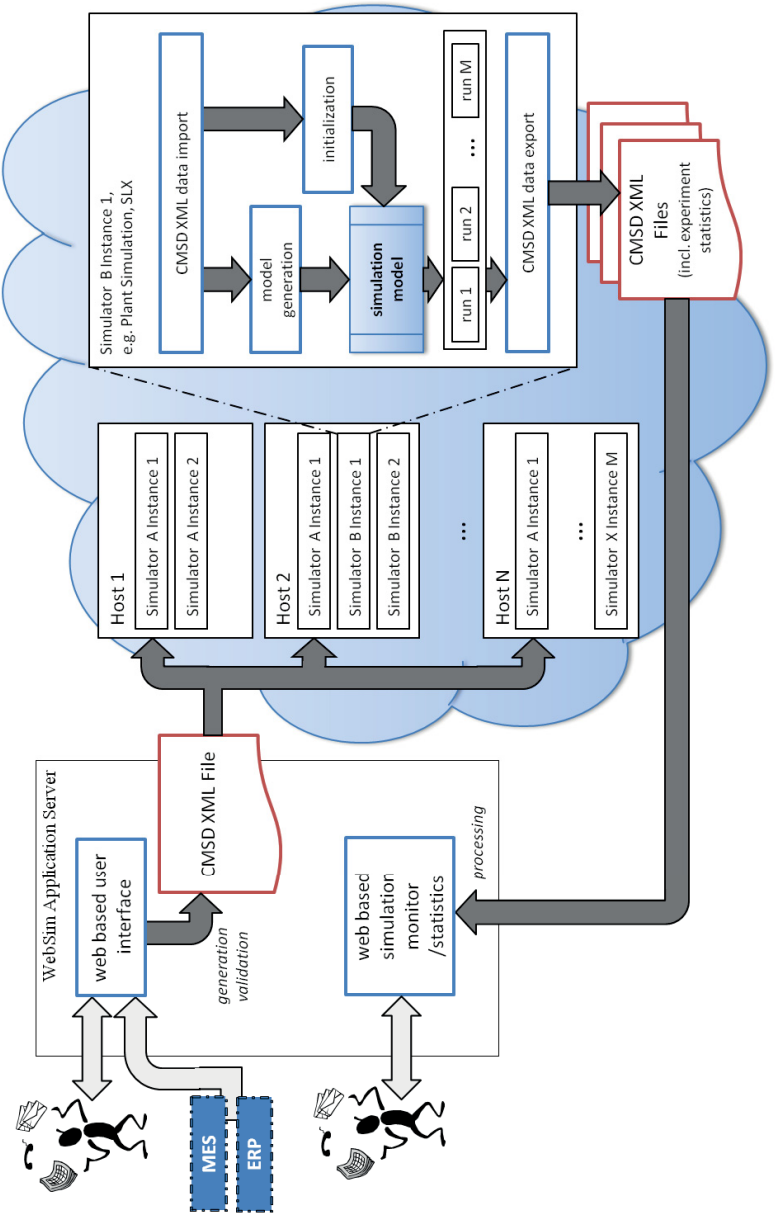


Abbildung 64: Beispielhafte Architektur des Frameworks

3.4.2 Modellgenerator und Initialisator

Allgemein innerhalb der gesamten Arbeit und im Speziellen im Framework stellen die CMSD basierten Modellgeneratoren das zentrale Herzstück dar. Im Folgenden wird wenn nicht explizit aufgeführt nicht zwischen Generatoren und Initialisatoren unterschieden, vielmehr wird die Annahme getroffen, dass alle Prototypen sowohl über die Möglichkeit zur Modellerzeugung als auch zur Modellinitialisierung, Durchführung von Simulationsläufen und der CMSD basierten Ergebnisspeicherung verfügen (vgl. Abbildung 65). Aus diesem Grund wird im Folgenden der Begriff Modellgenerator synonym verwendet.

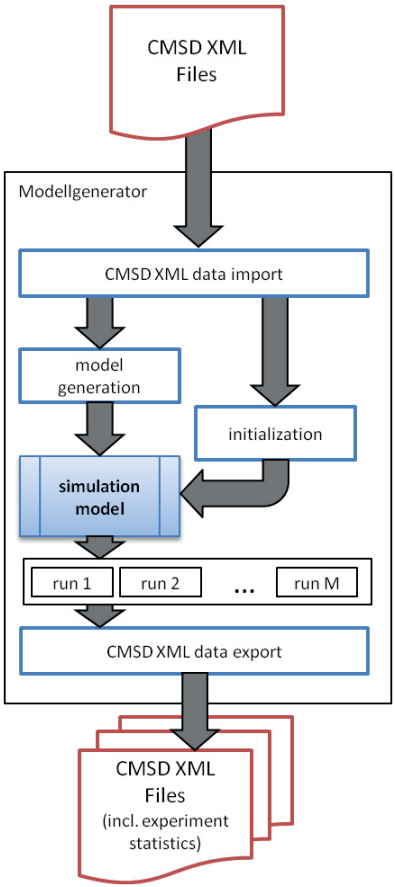


Abbildung 65: Grundfunktion (Generierung, Initialisierung, Simulation und Ergebnisspeicherung) aller CMSD basierten Modellgeneratoren

Die zentrale Grundanforderung an jeglichen CMSD basierten Modellgenerator ist, dass sämtliche Daten, d.h. sowohl Eingangs- als auch Ausgangsdaten, in Form von CMSD XML Dateien gestaltet sind, wobei manuelle Anpassungen des Modells, im Sinne der manuellen Adaption (vgl. Abschnitt 2.2.3) des Simulationsmodells, jederzeit möglich sind. Diese Änderungen müssen auch nach ggf. weiteren manuellen oder auf Basis veränderter CMSD Daten angestoßenen (Konzeptmodell) Adaptionsrunden erhalten bleiben. Der minimale Funktionsumfang, d.h. welche Systeme wie abbildbar sind, wird durch die CMSD Interpretation (vgl. Abschnitt 3.3) bestimmt. Zu beachten ist, dass diese jederzeit erweitert werden kann und ggf. in allen Modellgeneratoren zu Implementierungsaufwand führt. Empfehlenswert ist in den Modellgeneratoren die jeweilige Version der CMSD Interpretation anzugeben, die im Rahmen dieser Arbeit dokumentierte Version wird als V1.0 definiert. Weitere allgemeine Anforderungen an Modellgeneratoren wurden bereits in Abschnitt 2.2 definiert, besonders soll hier die Beachtung der GoM (vgl. Abschnitt 2.1.4) hervorgehoben werden. Besondere Relevanz für die technische Komponente Modellgenerator hat insbesondere der Grundsatz der Richtigkeit, daneben sind die Grundsätze Klarheit, Vergleichbarkeit und systematischen Aufbau von Bedeutung. Die Einhaltung der GoM stellt somit sicher, dass jeder Modellgenerator nicht nur "funktionierende" Modelle, die im jeweiligen Simulator lauffähig sind, erstellt sondern dass jeder Modellgenerator zu vergleichbaren Simulationsmodellen gelangt, welche dem modellierten CMSD-Konzeptmodell entsprechen. Des Weiteren sollen alle erzeugten Modelle so klar und systematisch gestaltet sein, dass eine Interpretation und manuelle Erweiterung durch Simulationsexperten ohne erheblichen Einarbeitungsaufwand möglich ist.

Im Folgenden werden zwei Implementierungsansätze für CMSD basierte Modellgeneratoren anhand je eines Prototyps vorgestellt. Die Ansätze unterscheiden sich allein dahingehend, wie das Modell erzeugt wird. So ist die Nutzung beider Ansätze bzw. sogar des konkreten Modellgenerators (Ansatz + Zielsimulator) für den Simulationsendnutzer transparent, ggf. unterschiedliche Funktionsumfänge sind nicht auf den Ansatz sondern allein auf unterschiedliche Entwicklungsstände zurückzuführen.

Der erste der im Folgenden näher beschriebenen Ansätze wird nachfolgend als interner Ansatz oder auch interne Modellgenerierung (vgl. Abschnitt 3.4.2.1) bezeichnet und stellt den "klassischen" Ansatz der Modellgenerierung dar (vgl. Abschnitt 2.2.1). Im Zuge der internen Modellgenerierung kommt eine im Zielsimulator implementierte Komponente zum Einsatz, die das CMSD Dokument verarbeitet und auf dessen Basis entsprechend das Simulationsmodell erzeugt, initialisiert, ausführt und Ergebnisse als interpretationskonforme CMSD Datei abspeichert.

Im Gegensatz dazu werden im zweiten Ansatz, nachfolgend als externer Ansatz oder auch externe Modellgenerierung bezeichnet (vgl. Abschnitt 3.4.2.2), keine speziellen

Komponenten im Simulator implementiert, sondern der Quellcode[30] des Modells direkt erzeugt. Bei der Erzeugung muss ebenfalls die Initialisierung erfolgen sowie Funktionen zur Ergebnisspeicherung vorgesehen werden. Der eigentliche Simulationslauf erfolgt im Simulator, dem der Quellcode übergeben wird. Diese Transformation von XML in Quellcode kann auf unterschiedliche Art und Weise realisiert werden. Im Rahmen dieser Arbeit wurde mit der eXtensible Stylesheet Language Transformation (XSLT) ein rein XML basierter Ansatz gewählt, der zudem allein auf der Anwendung deskriptiven Regeln basiert.

Teile bzw. Vorversionen der hier beschriebenen Prototypen wurden im Rahmen verschiedener Publikationen bereits veröffentlicht, vgl. [BS2010a, Fi2010, BSS2011a, BSS2011b, Be2012, BSS2012, Wü2013].

3.4.2.1 Prototyp 1 - interne CMSD-MG innerhalb eines bausteinorientierten Simulators (Plant Simulation)

Wie im letzten Abschnitt bereits angedeutet, stellt die interne Modellgenerierung den klassischen Ansatz der Modellgenerierung dar. Neuartig ist in dieser Arbeit aber die konsequente Nutzung eines Standards, konkret CMSD, für Ein- und Ausgangsdaten. Neben der üblichen Fokussierung auf die Modellerstellung allein wird eine umfängliche Betrachtung weiterer Aufgabenbereiche der Simulation vorgenommen, hierzu zählen die ebenso CMSD basierte Modellinitialisierung sowie die Ergebniserfassung und -ausgabe bzw. -speicherung.

Für den Prototyp wurde der in der Praxis verbreitete Simulator Plant Simulation[31] der Firma Siemens PLM Software gewählt. Plant Simulation stellt einen typischen Vertreter bausteinorientierter Simulatoren dar, der sich gerade in der Automobilindustrie großer Beliebtheit erfreut. Umfangreiche Erweiterungen können in der simulatorspezifischen, objektorientierten Scriptsprache SimTalk implementiert werden [Ba2008]. Trotz der für die Implementierung eines Prototypen nötigen Wahl eines speziellen Simulators kann das Konzept als allgemeingültig angesehen werden. Allein einige wenige Mindestvorrausetzungen müssen potentielle Zielsimulatoren erfüllen. So müssen Möglichkeiten zur Entwicklung eigener Programme, Komponenten und Bausteine bestehen sowie Bausteine/Komponenten dynamisch anzulegen und zu parametrieren sein, von Vorteil sind hierbei Möglichkeiten zur Objektorientierung. Zudem muss es, zur Steuerung der Modellgenerierung und somit zur Integration in das Gesamtframework, möglich sein, die einzelnen Komponenten/Scripte/Funktionen von außerhalb des

[30] "Unter dem Begriff Quelltext, auch Quellcode [..] wird in der Informatik der für Menschen lesbare, in einer Programmiersprache geschriebene Text eines Computerprogrammes [hier im Speziellen eines Simulationsmodells] verstanden" [WIKI2013e].
[31] Plant Simulation, in Version 10 [Si2013]

Simulators aufzurufen. Schließlich müssen zwingend XML Dateien verarbeitbar sein, d.h. es müssen XML Dateien auswertbar aber auch erzeugbar bzw. manipulierbar sein.

Wie bereits angerissen, wird im Rahmen des internen Modellgenerierungsansatzes eine im Simulator entwickelte Komponente genutzt. In den entwickelten Plant Simulation Prototypen wurden hierzu eine Vielzahl von Teilkomponenten wie bspw. Bausteinklassen, Methoden, Datenstrukturen (Tabellen, Variablen), Dialoge erstellt bzw. erweitert (vgl. Abbildung 66). Diese Softwareartefakte lassen sich grob in zwei Klassen einteilen. Zum einen in Teilkomponenten, welche als Bausteine (z.B. eine Einzelstation, Werker, BDE-Tabelle) oder Teile von diesen (z.B. hinterlegte Steuerungsmethoden) im zu erstellenden Simulationsmodell direkt genutzt werden, diese bilden die CMSD spezifische Klassenbibliothek (vgl. Abbildung 66, links). Zum anderen in Teilkomponenten, welche im Zuge der Modellgenerierung, -initialisierung, Ergebnisverarbeitung und allen damit verbundenen Prozessen eingesetzt werden. Diese Teilkomponenten sind in einem eigenen Netzwerk (CMSD_Schnittstelle) zusammengefasst (vgl. Abbildung 66, rechts). Innerhalb der CMSD_Schnittstelle sind verschiedene Schichten (in Abbildung 66 farblich hervorgehoben) zu identifizieren. In jeder Schicht sind wiederum einzelne ggf. aufeinander aufbauende Funktionalitäten implementiert.

In der ersten Schicht (blau hinterlegt) ist zum einen die Steuerungslogik des Modellgenerators (Dialog) an sich sowie der Datenimport implementiert. Der Datenimport besteht hierbei aus Methoden zum Parsen von CMSD XML Dateien. Konkret sind vier Methoden (Import_Ressourcen, Import_Arbeitsplan, Import_Aufträge und Import_Schichtkalender) implementiert, welche die im Generator angegebene CMSD Datei (Pfad_Eingabedatei) auswerten und die darin enthaltenen relevanten Daten in entsprechende Datenstrukturen der Datenhaltungsschicht (rot hinterlegt) speichern, fehlende Daten werden hierbei auf Defaultwerte gesetzt (vgl. Abbildung 67, links).

In der Datenhaltungsschicht sind nach erfolgreichem Datenimport durch den Nutzer jederzeit Änderungen an allen Modellparametern möglich, ohne das zugrundeliegende CMSD Dokument adaptieren zu müssen.

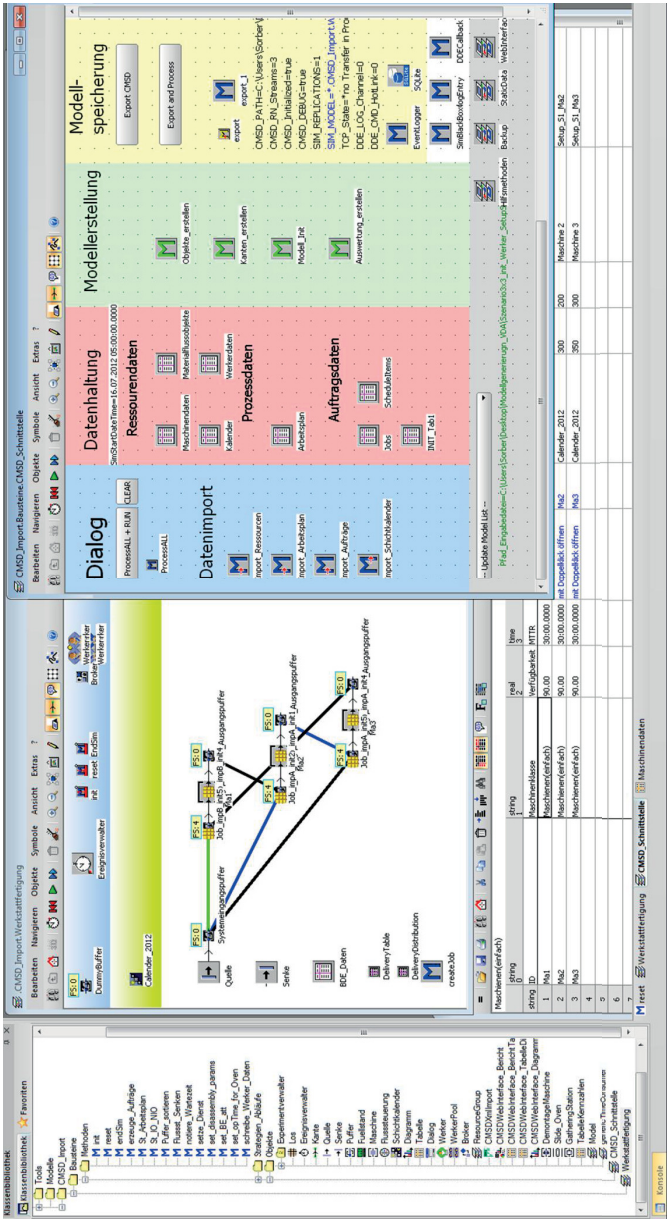


Abbildung 66: Screenshot des Plant Simulation Modellgeneratorprototypen

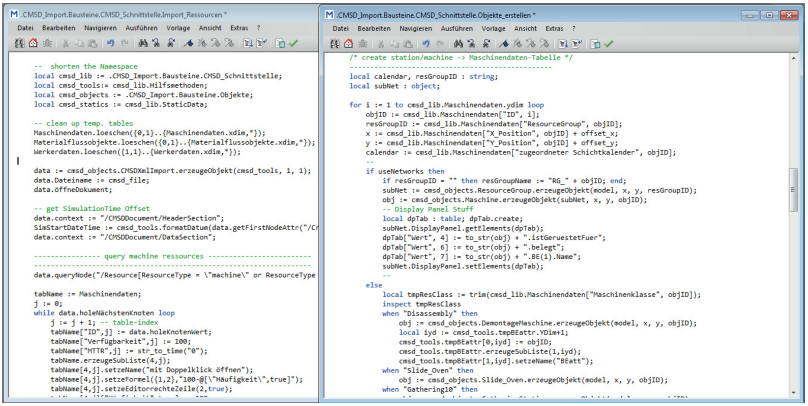


Abbildung 67: Auszug aus den Methoden Import_Ressourcen und Objekte_erstellen

Die dritte Schicht - Modellerstellung (grün hinterlegt) enthält vier aufeinander aufbauende Methoden, wobei die vierte Auswertung_erstellen optional bzw. veraltet ist und nur die Erstellung eines simulatorspezifischen, nicht CMSD basierten Auswertungsberichts ermöglicht. Zunächst wird durch die Methode Objekte_erstellen (vgl. Abbildung 67, rechts) wenn keine Modelladaption vorliegt (aus CMSD Dateien ableitbar, vgl. 3.3.10) ein entsprechendes Modellnetzwerk angelegt und sämtliche zwingend erforderlichen Standardobjekte (z.B. Ereignisverwalter) sowie alle in der Datenhaltungsschicht definierten Objekte (Stationen/Maschinen, Puffer, Schichtkalender, Werker usw.) erzeugt und parametriert. Im Fall von Adaption werden aufbauend auf dem Delta zwischen der originalen und der aktuellen CMSD Datei Änderungen an dem bestehenden Modell, z.B. Umsetzen einzelner Parameter, durchgeführt. Durch die Generatorsteuerung wird nach der erfolgreichen Abarbeitung der Methode Objekte_erstellen zwingend die Methode Kanten_erstellen aufgerufen, welche alle Verbindungen zwischen Maschinen/Stationen und den ihnen zugeordneten Puffern erzeugt, wenn diese nicht bereits vorhanden sind. Ergebnis der bisher skizzierten Schritte ist ein lauffähiges Modell, welches im Zuge der Abarbeitung der Methode Modell_init entsprechend initialisiert wird, d.h. es werden vor allem angefangene Aufträge (BEs) erzeugt und positioniert sowie Zeiten (z.B. Restbearbeitungszeiten), Zustände (z.B. gestört) und Rüstzustände aller Objekte gesetzt. Ebenfalls werden (wenn vorhanden) Werker ggf. laufenden Tätigkeiten zugeordnet. Schließlich werden im Rahmen der Initialisierung relevante Ereignisse in der internen BDE Datentabelle (vgl. Abbildung 68) angelegt, z.B. Freigabetermin eines begonnen Auftrags (vgl. Abschnitt 3.3.8 und 3.3.9 sowie Abschnitt 2.2.2).

Nach erfolgreicher Generierung und ggf. Initialisierung kann der Simulationslauf gestartet werden. Der Starttermin und die Dauer der Simulation werden hierbei ebenfalls dem CMSD Dokument entnommen, können aber auch über die externe Schnittstelle [32] überschrieben werden. Während des Simulationslaufs werden Informationen bzgl. Arbeitsplänen, der zu erzeugenden Aufträge usw. direkt aus der Datenhaltungsschicht heraus genutzt. Alle Ereignisse (z.B. Start einer Bearbeitung) werden als BDE ähnliche Datensätze (vgl. Abschnitt 3.3.9) in der Tabelle BDE_Daten gespeichert (vgl. Abbildung 68).

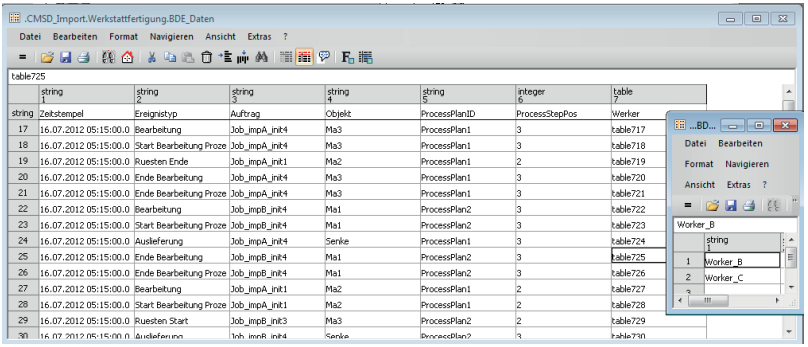


Abbildung 68: Screenshot der internen BDE-Datentabelle des Plant Simulation Prototypen

Jederzeit, aber am sinnvollsten nach Beenden des Simulationslaufs, kann das Modell und dessen Zustand gespeichert und in CMSD exportiert werden (vgl. Abschnitt 3.3.9). Alle nötigen Methoden und sonstigen Objekte sind in der vierten Schicht - Modellspeicherung (gelb hinterlegt) eingeordnet. Dazu werden zum einen die aktuellen Zustände der Objekte des Simulationsmodells analysiert und zum anderen die Daten der BDE_Daten Tabelle herangezogen.

Abgerundet wird der Plant Simulation Modellgenerator durch eine Vielzahl von Hilfsfunktionen (grau hinterlegt). So sind Methoden implementiert, welche in anderen Komponenten genutzt werden, z.B. Methoden zum Umwandeln von CMSD Datumsangaben in durch den Simulator interpretierbare Werte (und umgekehrt). Des Weiteren sind Methoden implementiert, die es erlauben alle Prozesse des Prototypen über eine externe HTTP-Schnittstelle bzw. über Übergabeparameter beim Start des Simulators zu steuern. Über diese Schnittstellen kann sowohl der Pfad der Eingangs- und Ausgangs-CMSD Datei gesetzt werden, die Modellgenerierung und -initialisierung gestartet, Simulationsläufe angewiesen als auch der Datenexport angestoßen werden.

[32] die externe Schnittstelle wird in diesem Abschnitt an geeigneter Stelle separat beschrieben.

Ansätze zu ersten Versionen des Prototyps wurden bereits in [BS2010a] [Fi2010] [BSS2011b] publiziert, die Initialisierung wurde in [BSS2011a] und die Ergebnisdatenspeicherung in [BSS2012] aufbauend separat thematisiert.

3.4.2.2 Prototyp 2 - externe CMSD-MG innerhalb eines sprachenbasierten Simulators (SLX)

Der zweite neuartigere Ansatz zur Modellgenerierung wird vom Autor als externe Modellgenerierung bezeichnet. Wie bereits angerissen, ist die Grundidee abweichend vom internen Ansatz darin zu sehen, dass nicht aus dem Simulator heraus CMSD Daten interpretiert werden, sondern dass auf Basis von CMSD Daten durch eine zusätzliche Softwarekomponente Simulationsquellecode erzeugt wird[33].

Prinzipiell ist dieser Ansatz für alle Simulatoren anwendbar, bei denen sich allein mit Quellcode Modelle beschreiben lassen und die die Möglichkeit haben, extern erzeugten Code zur Ausführung zu bringen, z.B. indem komplette Programme in Form von Quellcodedaten geladen und gestartet werden. Für den hier dokumentierten Prototypen wurde der sprachbasierte, objektorientierte Simulator, Simulation Language with eXtensibilities (SLX) in Version 2.0 [He1999, SH2002] sowie das Visualisierungswerkzeug desselben Herstellers Proof Animation [He1997] gewählt (vgl. Abbildung 69). Proof Animation kann sowohl zur Online- als auch zur Postsimulationsvisualisierung genutzt werden. Für den Prototyp wird keine Onlinevisualisierung implementiert. Positiv für die externe Modellgenerierung ist, dass auch Proof Animation Visualisierung allein auf Quellcode basiert und z.B. keine zusätzlichen Binärdaten wie Grafiken zwingend benötigt. So sind für eine Visualisierung zwei Dateien zu erzeugen, zum einen eine Layoutdatei (.lay) zum anderen eine Animation Trace Datei (.atf). Die Layoutdatei beinhaltet Informationen zu:

- den Geometrien und Aussehen der Modellobjekte (z.B. Stationen),
- Bauteilen und Klassen, als bewegliche Elemente,
- Views, mittels derer Teilbereiche definiert werden und
- Diagrammen, Texten, Nachrichten und weiteren Anzeigeobjekten.

[33] Erste Prototypen entstanden in studentischen Arbeiten [Wü2012] und [Wü2013].

Neben der das System beschreibenden Layoutdatei wird eine Animation Trace Datei[34] des zu visualisierenden Simulationslaufs benötigt, diese enthält alle animationsrelevanten Ereignisse.

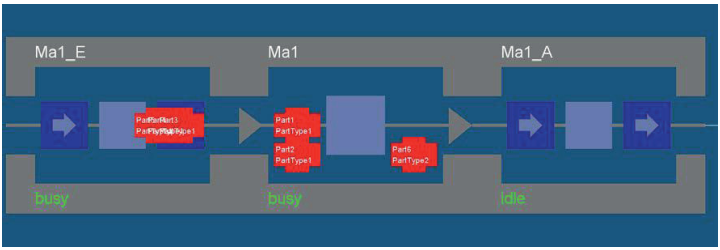


Abbildung 69: Screenshot eines Ausschnitts aus einer Proof Animation Visualisierung, hier eine Station mit Ein- und Ausgangspuffer [Be2012, S. 9]

Zur Erzeugung des Quellcodes kommen theoretisch eine Vielzahl möglicher Technologien bzw. Varianten in Betracht, so könnten bspw. eigens entwickelte Wrapper diese Aufgabe übernehmen. Im Rahmen dieser Arbeit wird angestrebt, eine auf Standards bzw. Standardtechnologien basierende Lösung zu nutzen, die flexibel bzgl. z.B. Erweiterungen des CMSD Standards und der CMSD Interpretation ist, den Entwicklungsaufwand so gering wie möglich hält und trotzdem robust und etabliert ist. Unter diesen Gesichtspunkten und da die CMSD Dokumente in Form von XML Dateien vorliegen, wurde XSLT als Technologie gewählt.

eXtensible Stylesheet Language Transformation - XSLT

Die eXtensible Stylesheet Language Transformation oder XML Stylesheet Transformation ist eine deklarative "Programmiersprache" zur Umwandlung von XML Dokumenten bzw. zur Transformation von XML Bäumen. XSLT ist Teil der Extensible Stylesheet Language (XSL) einer in XML notierten Familie von Transformationssprachen (vgl. [WIKI2013f], [W3C2013a]). Aktuell ist XSLT Version 2.0 (vgl. [W3C2013b]), welche eine erhebliche Erweiterung gegenüber XSLT 1.0 darstellt. Erst die mit Version 2.0 eingeführten Erweiterungen ermöglichen die hier beschriebene Modellgenerierung. Im Folgenden wird mit den Begriff XSLT immer XSLT 2.0 verstanden.

[34] Im Fall von Onlinevisualisierung wird statt der Animation Trace Datei eine Animation Trace Stream (Datenstrom) genutzt [He1997].

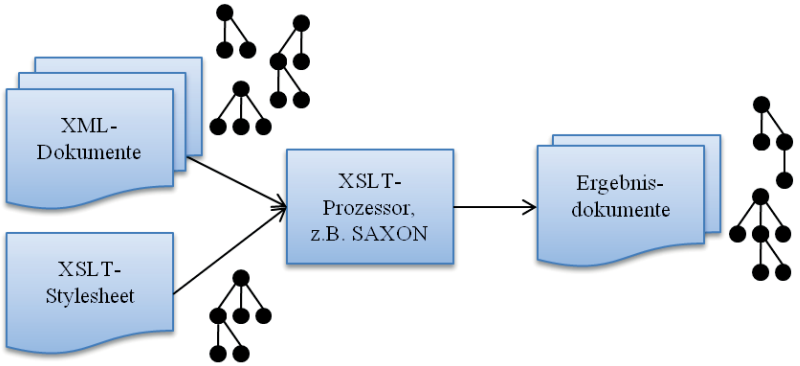
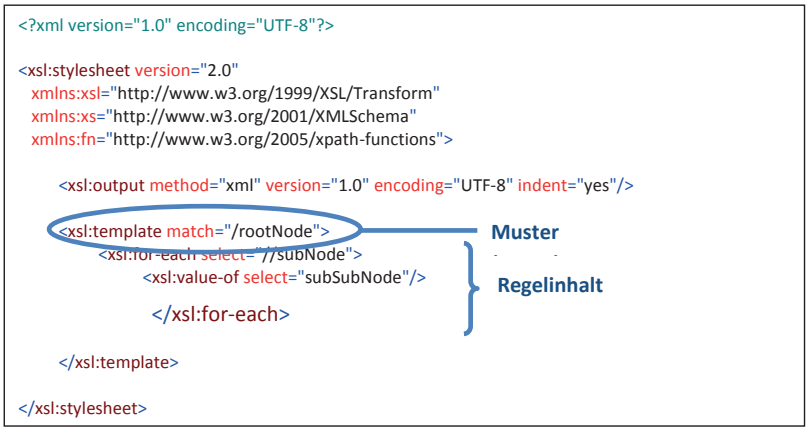


Abbildung 70: Ablauf der eXtensible Stylesheet Language Transformation (XSLT) (in Anlehnung an [Be2012, S. 3])

XSLT ermöglicht im Rahmen eines Transformationsprozesses neue Dokumente zu erzeugen, die auf bestehenden XML Dokumenten und einem XSLT Stylesheet, welches ein Set von XSLT Regeln (XSLT rules) enthält, basieren (vgl. Abbildung 70). Im Rahmen der Transformation werden die Originaldokumente nicht verändert sondern durch einen XSLT Prozessor Ergebnisdokumente neu erzeugt. Die Ergebnisdokumente können wiederum XML Dokumente (ggf. mit eigenem Schema) oder auch Dokumente in anderen Formaten, z.B. HTML, Fließtext oder gar Binärcode sein. Die XSLT Grundsätze sind, dass:

- XSLT auf den logischen Baumstrukturen der XML-Dokumente aufbaut,
- alle Transformationsregeln im XSLT-Stylesheet ebenfalls mittels XML definiert werden,
- für die Adressierung der einzelnen Unterstrukturen der XML Bäume im Zuge der Transformation XPath 2.0 (vgl. [W3C2013c]) verwendet wird und
- ein XSLT-Prozessor (z.B. Saxon [Sa2013]) zur Verarbeitung der XSLT Stylesheets und Quelldokumente verwendet wird.

Die Transformationsregeln, auch Templates genannt, bestehen aus einem Pattern (Muster), welches mittels XPath beschreibt für welches XML Element (auch Teilbaum) die Regel gilt und aus dem Content (Inhalt), welcher das zu erzeugende Ergebnis beschreibt (vgl. Abbildung 71).




```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/rootNode">
        <xsl:for-each select="//subNode">
            <xsl:value-of select="subSubNode"/>
        </xsl:for-each>

    </xsl:template>

</xsl:stylesheet>
```

Abbildung 71: Beispiel einer simplen XSLT Regel (in Anlehnung an [Be2012, S. 4])

Der Inhaltsblock der Regeln kann verschiedene Elemente beinhalten, von einfachem Text bis hin zu mächtigen Funktionen zur Analyse und Transformation von Daten der Quelldokumente. Auch können weitere Kontrollstrukturen (Schleifen, Verzweigungen) und sogar Aufrufe externer Funktionalitäten (z.B. Java Methoden) erfolgen.

Im Folgenden wird der Einsatz von XSLT zur Generierung von SLX und Proof Animation Modellen vorgestellt. Prinzipiell sollen für alle in CMSD vorkommenden Entitäten entsprechende Abbildungen in SLX und ggf. in Proof Animation geschaffen werden. Hierzu werden im Rahmen des Prozesses SLX Klassen und deren Attribute basierend auf den CMSD Schemata gebildet, die Schemata werden dabei dynamisch ausgewertet. Dieses Vorgehen ermöglicht Änderungen am Standard aber vor allem auch zusätzliche, entfernte oder veränderte Properties als Attribut in den entsprechenden Klassen der SLX Klassenbibliothek abzubilden. Neben den Attributen und deren Typen sind dabei besonders Referenzen zwischen einzelnen Entitäten von Bedeutung. Um Referenzen korrekt abzubilden, ist zwischen eindeutig identifizierbaren Objekten (UniqueEntities), in Teilbereichen eindeutig identifizierbaren Objekten (LimitedUniqueEntities), Objekten ohne Identifikator und den Spezialfall der Referenzen (Erben von AbstractEntityReference) zu unterscheiden. Des Weiteren wurden über die reine CMSD Beschreibung hinaus Spezialisierungen der Klasse Ressource in Abhängigkeit vom RessourceType vorab definiert, d.h. konkret wurden die Klassen für Maschine, Puffer und Werker abgeleitet. Dies ist nötig, um das unterschiedliche Verhalten der Ressourcentypen zu realisieren. Zur Abbildung des Verhaltens wurden die Klasse Machine, Buffer, globalWorkerPool sowie Shift und Job manuell erweitert, die Logik wurde in Form von Actions-Methoden in der CMSD SLX Klassenbibliothek implementiert [Wü2013, S.107ff].

Der Ablauf der externen Modellgenerierung von der Modellerstellung bis zur Ergebnispräsentation lässt sich grob in vier Schritte unterteilen (vgl. Abbildung 72). Im ersten Schritt werden mittels eines XSLT Prozessors die Quell-CMSD-Dateien in ein, bzw. wenn eine Visualisierung gewünscht ist, in zwei Ergebnisdateien transformiert. Die erste Ergebnisdatei beinhaltet SLX Quellcode (1a) und die zweite optionale Datei die Proof Animation Layoutdaten. Die Transformation basiert auf zwei XSLT Stylesheetdateien, welche die Umwandlungsregeln zu SLX Quelltext (CMSD_To_SLX.xslt, vgl. Abbildung 74) und das Datenformat von Proof Animation für Layoutdaten (CMSD_TO_P5.xslt, vgl. Abbildung 73) beinhalten.

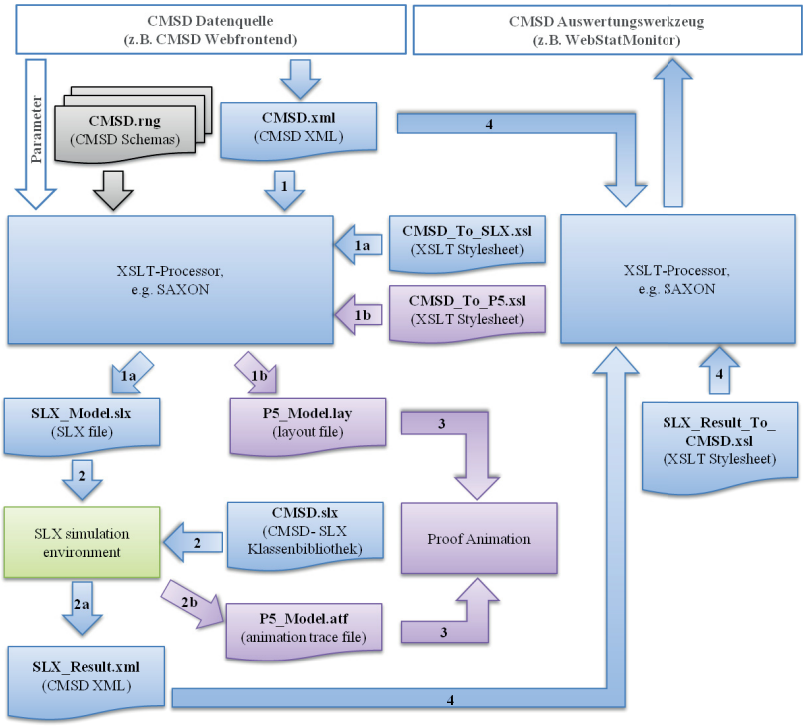


Abbildung 72: Ablauf der externen Modellgenerierung am Beispiel SLX und Proof Animation (in Anlehnung an [Be2012, S. 6; Wü2013, S. 99])

```
...
<!-- LayoutObjekte -->
<xsl:call-template name="CreateLayoutObject">
        <xsl:with-param name="Identifier" select="$ResourceIdentifier"/>
        <xsl:with-param name="StartX" select="$LocationX"/>
        <xsl:with-param name="StartY" select="$LocationY"/>
        <xsl:with-param name="Type" select="string('Machine')"/>
</xsl:call-template>
...
<!-- Bearbeitungsstationen-->
<xsl:template name="CreateMachineClass">
        <xsl:text>Color L4</xsl:text><xsl:value-of select="$newline"/>
        <xsl:text>Define Class Machine</xsl:text><xsl:value-of select="$newline"/>
...
```

Abbildung 73: Auszug des Stylesheets CMSD_To_P5.xslt

```
...
<xsl:variable name="ClassReference">
    <xsl:choose>
        <!-- Erstellung von Unique und LimitedUniqueEntities -->
        <xsl:when test="$CMSDClasses/class[name = $ClassName]/inherit = 'UniqueEntity' or
                $CMSDClasses/class[name = $ClassName]/inherit = 'LimitedUniqueEntity'">
            <xsl:value-of select="./Identifier"/>
        </xsl:when>
        <!-- Erstellung einer Property -->
        <xsl:when test="$ClassName = 'Property'">
            <xsl:choose>
                <xsl:when test="not(./Distribution) and not(./Value)">
                    <xsl:text>ReferenceProperty</xsl:text>
                </xsl:when>
                <xsl:when test="string(number(./Value)) != 'NaN'">
                    <xsl:text>SimpleNumericProperty</xsl:text>
                </xsl:when>
                 <xsl:otherwise>
                        <xsl:text>SimpleTextProperty</xsl:text>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="$ClassName"/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:variable>
<!-- Konstruktor -->
<xsl:value-of select="concat($ClassReference, ' = new ')"/>
...
```

Abbildung 74: Auszug des Stylesheets CMSD_To_SLX.xslt

Zusätzlich werden bei der Erzeugung desSLX Codes die CMSD Schemata wie angerissen zur Erzeugung (Bestimmung der Klasse, der Attribute und Datentypen sowie Beziehungen) der SLX Datei herangezogen. Ergebnis des ersten Schrittes ist zum einen eine SLX Datei (SLX_Model.slx), in welcher alle nötigen Objekte sowie die Attributausprägungen codiert sind (vgl. Abbildung 75), zum anderen eine Layoutdatei (P5_Model.lay), welche als Basis der Visualisierung mit Proof Animation genutzt werden kann (vgl. Abbildung 76). Die SLX Datei selbst beinhaltet weder Klassendefinitionen noch implementierte Methoden zur Steuerung, sondern nutzt die separat gekapselte CMSD SLX Klassenbibliothek CMSD.slx, welche alle nötigen Implementierungen beinhaltet (vgl. Abbildung 77).

```
...pointer(Resource) Ma1;
pointer(Resource) Ma2;
...
pointer(Resource) Ma1_Eingangspuffer;
...
pointer(LayoutObject) Lay_Ma1;
...
Lay_Ma1 = new LayoutObject("Lay_Ma1");
Lay_Ma1->m_Description = "Layout Element Ma1";
...
Ma1 = new Machine("Ma1");
Ma1->m_Description = "Maschine 1";
Ma1->m_ResourceType = machine;
...
SetupDefinitionReference = new SetupDefinitionReference();
SetupDefinitionReference->m_SetupDefinitionIdentifier = "SetupDefinition_abgeruestet_0";
Ma1->m_CurrentSetup = SetupDefinitionReference;...
```

Abbildung 75: Auszug einer automatisch erzeugten SLX Datei (SLX_Model.slx) - Ergebnis Schritt 1a

```
...Color L4
Define Class Machine
…
Fill 2.1516 -0.3169
Line -5 -5 -5 5
Line 5 -5 -5 -5
Line 5 5 5 -5
Line -5 5 5 5
…
Message MachineStatus 1 2 LJ -15 -13.5 BG Backdrop disassemble
Message MachineIdentifier 1 2 LJ -15 11.5 BG Backdrop Machine Name
…
CPO Buffer Ma3_E 298 350
CPO Machine Ma3 338 350
CPO Buffer Ma3_A 378 350 ...
```

Abbildung 76: Auszug einer automatisch erzeugten Proof Animation Layoutdatei (P5_Model.lay) - Ergebnis Schritt 1b

```
...
    public class Machine(string(*) identifier) subclass(Resource(identifier)) {
        private control boolean m_Working;
        ...
        private double m_MTTR;
        private double m_MTBF;
        private rn_stream m_MTBF_Stream;
        private rn_stream m_MTTR_Stream;

        initial {...}

        actions {...}
        ...
        public method setup()  {...}

        public method work(set(PartGroup) partsToConsume, set(PartGroup) partsToProduce)
        {
            pointer(SkillReference) skillReference;
            ...
            writeEvent("Bearbeitung", TRUE);
            ...
        }
        ...
    }
...
```

Abbildung 77: Auszug aus der CMSD SLX Klassenbibliothek (CMSD.slx)

Im zweiten Schritt wird das erzeugte SLX Modell in Verbindung mit der Klassenbibliothek (CMSD.slx) zur Ausführung gebracht. Die Klassen der Klassenbibliothek beinhalten neben der angesprochenen Steuerungslogik auch alle Funktionen zur Speicherung der Ereignisdaten (Schritt 2a; SLX_Result.xml; vgl. Abschnitt 3.3.9) sowie der Animation-Trace-Daten (Schritt 2b; P5_Modell.atf, vgl. Abbildung 78), welche zur Visualisierung benötigt werden. Hierdurch wird sichergestellt, dass diese während der Ausführung des Modells in der SLX Laufzeitumgebung alle gewünschten Daten entsprechend erzeugt, wobei zu bemerken ist, dass mittels eines Parameters die Erzeugung der Animation-Trace-Datei je nach Bedarf an- und abgestellt werden kann.

Ist eine Visualisierung gewünscht, d.h. per Parameter aktiviert, kann diese optional als dritter Schritt erfolgen, alle benötigten Daten liegen bereits direkt vor. Dem automatisch gestarteten Proof Animation werden die entsprechenden Pfade zu der Layoutdatei aus Schritt 1 (P5_Model.lay) und der Animation-Trace-Datei aus Schritt 2 (P5_Model.atf, vgl. Abbildung 78) übergeben.

```
...
write MachineIdentifier(Ma3) Ma3
write MachineStatus(Ma3) idle
...
place Part3 at 308.000 350.000
write MachineStatus(Ma3) busy
...
```

Abbildung 78: Auszug aus einer Animation-Trace-Datei (P5_Model.atf)

Aus Performancegründen und um die Implementierung in SLX so einfach wie möglich zu halten, werden zwar Ergebnisdaten (SLX_Result.xml) automatisch erzeugt, diese sind aber nur Teilbäume einer CMSD Datei, konkret sind nur die veränderlichen Attribute bzw. Entitäten, z.B. bzgl. des Status und Rüstzustands der Stationen, den Status aller Aufträge und die Ereignisse (Events) enthalten. Um eine Auswertung oder Weiternutzung zu ermöglichen, müssen diese Daten mit den statischen Daten, z.B. Layoutinformationen, Werkerfähigkeiten, Rüstmatrizen usw. des Quell-CMSD-Modells zusammengesetzt werden. Auch in diesem vierten Schritt wird wiederum XSLT als Technologie genutzt, um auf Basis der Quell-CMSD-Datei, der Ergebnisdatei (SLX_Result.xml) und eines speziellen Stylesheets (SLX_Result_To_CMSD.xslt) eine vollständige, den Simulationsendzustand repräsentierende, interpretations- und standardkonforme CMSD Datei zu erzeugen. Diese Datei kann entsprechend weiter genutzt werden, eine Möglichkeit stellt der Import in den WebStatMonitor (vgl. Abschnitt 3.4.4) dar.

Weitere Details zur Implementierung können [Be2012] und [Wü2013, S. 96ff] entnommen werden.

3.4.3 CMSD Datenanreicherung und Validierung - das CMSD Webfrontend

Neben den Modellgeneratoren stellt das CMSD Webfrontend die zweite Kernkomponente des Frameworks dar. Das Webfrontend stellt dem Endnutzer eine Vielzahl von Funktionalitäten zur Durchführung von Simulationsstudien zur Verfügung und dient als die zentrale Steuerungskomponente des Frameworks. Ein erster und seit dem kontinuierlich und umfänglich weiterentwickelter Prototyp entstand im Rahmen einer studentischen Arbeit [Ho2011].

Das Webfrontend selbst ist eine ASP.Net Webanwendung, welche auf einem Server betrieben und mittels beliebiger aktueller Internetbrowser nutzbar ist. ASP.Net ist eine serverseitige Technologie von Microsoft zum Erstellen dynamischer Webseiten, Webanwendungen und Webservices auf Basis des Microsoft-.NET-Frameworks. Als Entwicklungsplattform dient Visual Studio 2010, als .Net-Programmiersprache wurde C# gewählt. Dies erlaubt eine vollständige objektorientierte Entwicklung und stellt umfangreiche Basisfunktionalitäten, z.B. für die Authentifizierung zur Verfügung [Sc2012; MSC2013].

Die Funktionen des Webfrontends lassen sich grob in die Bereiche:

- CMSD Datenpflege und -verwaltung, inklusive Schnittstellen zu CMSD Datenquellen, z.B. SAP ERP (vgl. [Wü2013])
- Simulationsteuerung und Experimentverwaltung sowie
- Administration, Nutzerverwaltung

einteilen.

Der Bereich CMSD Datenpflege und -verwaltung umfasst eine Sammlung von Funktionalitäten zum Erstellen, Importieren, Speichern, Laden, Adaptieren und Validieren von CMSD Daten. Hierzu werden verschiedene Webformulare zur Verfügung gestellt, welche erlauben alle benötigten CMSD Entitäten und deren Attribute anzulegen bzw. zu bearbeiten, eine grobe Sitemap[35] der Anwendung ist Abbildung 79 zu entnehmen. In Abbildung 80 ist das Einstiegsformular der Anwendung abgebildet, über welches alle Subwebformulare der Hauptentitäten z.B. der Ressourcen /Bearbeitungsstation (vgl. Abbildung 81), Werker oder Aufträge (Jobs) erreichbar sind. Mittels der Formulare lassen sich sowohl beliebig viele neue Entitäten anlegen als auch jederzeit bestehende verändern bzw. ergänzen.

[35] Als Sitemap wird in der Webentwicklung eine Übersicht der vorhandenen Seiten und die Navigationsmöglichkeiten verstanden.

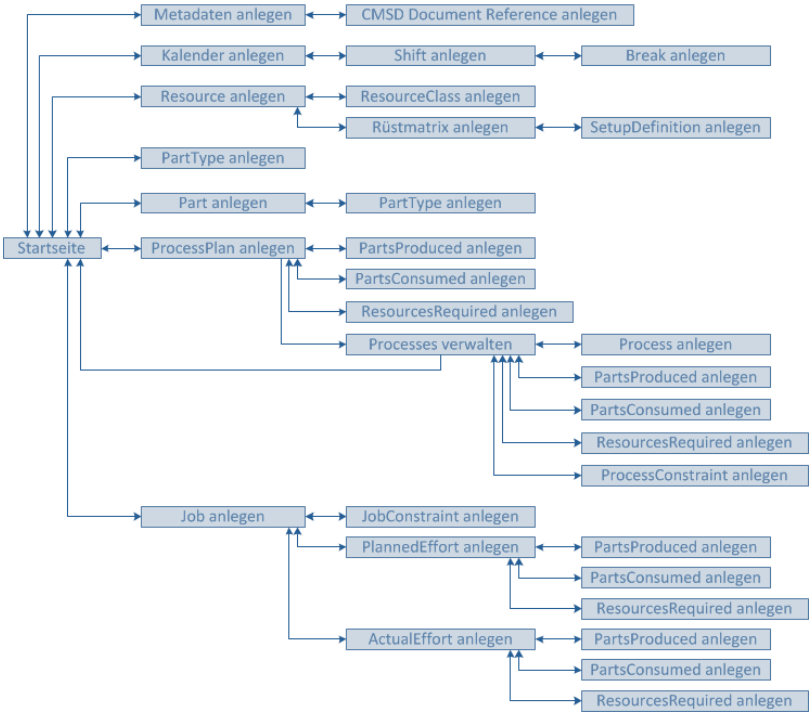


Abbildung 79: Ausschnitt der Sitemap des Webfrontends (in Anlehnung an [Ho2011])

In den Formularen der einzelnen Entitäten sind in der Regel weitere Subformulare von referenzierbaren Entitäten aufrufbar. So ist beispielsweise das Formular zum Anlegen (+) bzw. Pflegen (edit) von ResourceClass Entitäten aus dem Resource Formular heraus aufrufbar (vgl. Abbildung 81). Einmal angelegte Entitäten sind an allen Stellen, an denen Referenzen auf diese möglich sind, direkt über entsprechende Drop-Down-Listen zugänglich. Zu bemerken ist, dass für jede CMSD Entität ein Objekt existiert, aber keine eins zu eins Umsetzung der Entitäten in Webformulare erfolgt, in vielen Fällen werden mittels eines Formulars mehrere CMSD Entitäten angelegt bzw. gepflegt. So beinhaltet z.B. das in Abbildung 81 abgebildete Resource-Formular das gesamte Modellierungspattern, d.h. es werden sowohl die Bearbeitungsstation als auch die entsprechenden Ein- und Ausgangspuffer einbezogen. Ein weiteres Beispiel ist die ebenfalls in Abbildung 81 (rechts) erkennbare Rüstmatrix, welche die gleichzeitige Betrachtung aller SetupChangeoverDefinitions mit den verbundenen Rüstzuständen einer Station erlaubt, zusätzlich ist das Aufrufen des Formulars zum Anlegen bzw. Pflegen der Rüstzustände (Setup) möglich.

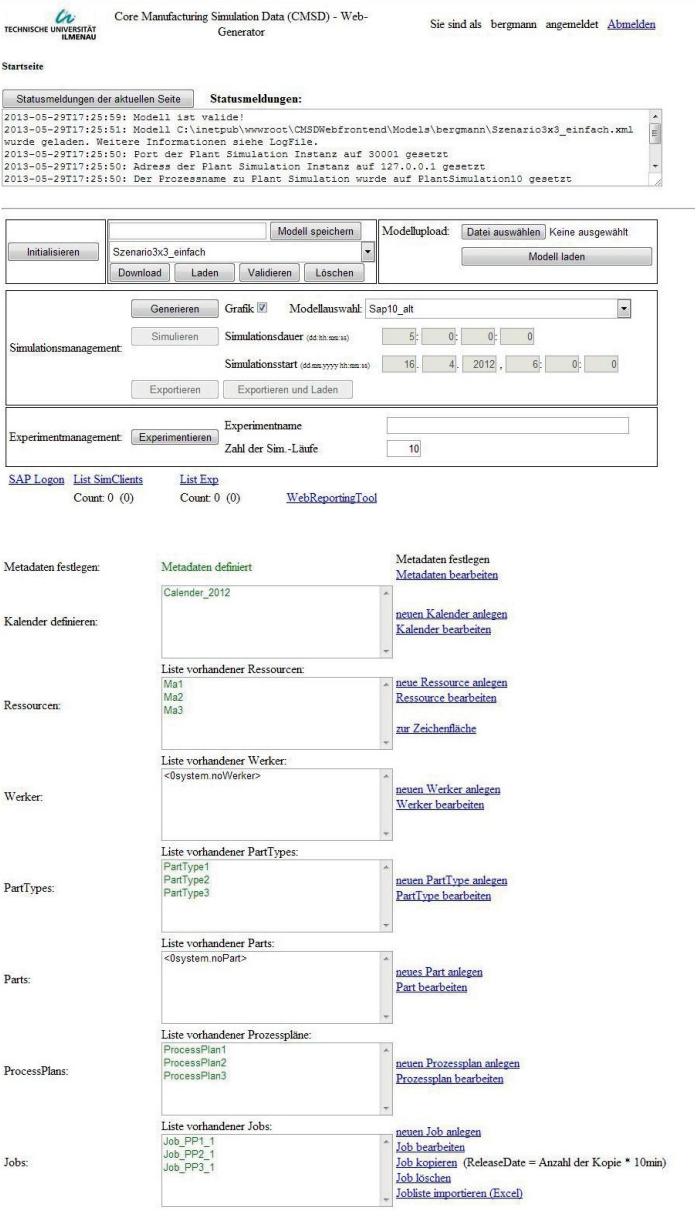


Abbildung 80: Screenshot des Einstiegsbildschirms des CMSD Webfrontends

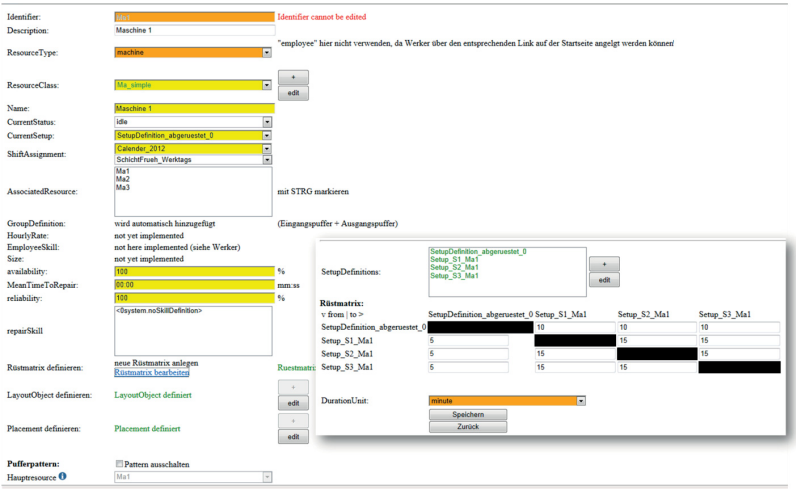


Abbildung 81: Screenshot der Eingabemaske einer Bearbeitungsstation sowie der zugehörigen Rüstmatrix

Neben dem auch für Aufträge (Jobs) verfügbaren Formular zum Anlegen und Bearbeiten entsprechender CMSD Entitäten, können Auftragsentitäten auch auf Basis eines Excel-Imports generiert werden (vgl. Tabelle 39).

Tabelle 39: Beispieltabelle für den Auftragsimport des Webfrontens

JobID	*Description*	*Status*	*Process PlanID*	*Release Date*	*DueDate*	*Aktueller PP Schritt*
Job_A_1	Job A1	released	ProcessPlan1	16.7.12 7:00	17.7.12 12:00	
Job_B_1	Job B1	released	ProcessPlan2	16.7.12 7:00	17.7.12 12:00	
Job_A_1 I	Job A1 init	started	ProcessPlan1	13.7.12 7:00	16.7.12 20:00	PP1_ProcessB

Anzugeben sind hierbei die wichtigsten Attribute der Jobs, dabei ist auf Konsistenz mit dem vorab definierten CMSD Modell Entitäten zu achten, z.B. bzgl. der Arbeitsplanbezeichnung (ProcessPlanID). Für angefangene Aufträge sind entsprechend Daten über den aktuellen Bearbeitungsstand aufzunehmen. Der Massenimport mittels Excel erleichtert die Anwendung des Webfrontends in vielen Anwendungsszenarien und ist gerade für Aufträge sinnvoll, da deren Zahl im Vergleich zu beispielsweise Bearbeitungsstationen oder Werkern sehr groß sein kann. Zudem wiesen Aufträge tendenziell die höchste Änderungshäufigkeit auf, so ist z.B. im Fall der betriebsbegleitenden Simulation mit dem Ziel Produktionsprogramme für bestimmte

Zeitabschnitte zu analysieren mit vielen verschiedenen Auftragsmixen bzw. verschiedenen zu betrachtenden Zeiträumen zu rechnen.

Des Weiteren sind in allen Formularen die im Rahmen der CMSD Interpretation definierten Pflichtfelder gekennzeichnet. Hierdurch wird der Nutzer zusätzlich für ggf. relevante Attribute sensibilisiert. Im Rahmen der Pflichtfelddefinition wurden fünf verschiedene Level, welche auch farblich von einander unterscheidbar sind, definiert. Aktuell sind mit Level 5 (Orange) alle im Standard definierten Pflichtfelder (meist nur die Identifier), mit Level 4 (Gelb) alle für die Modellgeneratoren aktuell zwingend benötigten Attribute, mit Level 3 (Khaki) alle für die Modellgenerierung empfohlenen Attribute gekennzeichnet, alle sonstigen Attribute sind mit Level 0 belegt. Level 1 und 2 haben aktuell keine Bedeutung. Die Definition der Pflichtfelder erfolgt in der jederzeit anpassbaren XML Datei RequiredElements.xml (vgl. Anhang D) innerhalb der Webanwendung.

Wie bereits angerissen werden alle Entitäten während der Bearbeitung in Objekten gehalten, dazu wurde für jeden in CMSD vorkommenden Entitätentyp eine entsprechende Klasse entwickelt. Die Objekte werden in Form einer Array-Liste im Sitzungszustand (Session State)[36] des Nutzers vorgehalten. Alle im Framework zur Abbildung der CMSD Entitäten angelegten Klassen erben hierbei von der Klasse XElement. Diese ist wiederum eine Klasse der LINQ[37] to XML (Language Integrated Query für XML) und dient der Abbildung von XML-Elementen. Jede dieser XML-Elementarklassen kennzeichnet sich durch einen Tag-Namen sowie dessen Inhalt, welcher wiederum aus XML-Elementen bestehen kann [DG2010, S.313, 424f]. Dieses Vorgehen erlaubt, unter Nutzung der LINQ to XML Basisfunktionalitäten, einfaches Speichern, Validieren gegen Schemata und Laden der Strukturen von CMSD XML Dateien. So werden erst bei der Speicherung des gesamten Modells unter ggf. Angabe eines Modellnamens (vgl. Abbildung 80, Modell Speichern) entsprechende CMSD XML Dateien erzeugt. Diese werden in einem für jeden Nutzer separat definierbarem Verzeichnis abgelegt, und können jederzeit mittels einer Drop-Down-Liste ausgewählt, geladen oder gelöscht werden. Zudem ist es jederzeit möglich, auch lokale CMSD Dateien zu laden (vgl. Abbildung 80, Modellupload) bzw. CMSD Dateien lokal zu speichern (vgl. Abbildung 80, Download). Diese Funktionalität erlaubt somit jederzeit das Laden von CMSD Daten aus beliebigen Quellsystemen. Als Schnittstelle ist zum einen eine einfache SAP ERP Schnittstelle implementiert, die mittels Webservices das

[36] Mittels des ASP.Net Sitzungszustands wird die Zustandslosigkeit des HTTP Protokolls umgangen, indem im Sitzungszustand Werte für einen Benutzer gespeichert und abgerufen werden können, während dieser durch die ASP.NET-Seiten einer Webanwendung navigiert [MSDN2013a].

[37] "LINQ (Abkürzung für Language Integrated Query [...]) ist ein programmtechnisches Verfahren von Microsoft zum Zugriff auf Daten" [WIKI2013g].

Auslesen einzelner Entitäten bzw. deren Attribute direkt aus den entsprechenden Webformularen erlaubt. Zum anderen wurde eine umfangreichere SAP ERP Schnittstelle implementiert, welche vollständige CMSD Dateien erzeugt, diese ist in [Wü2013] ausführlich beschrieben. Zu bemerken ist, dass jederzeit manuell angestoßen (vgl. Abbildung 80, Validieren) sowie vor jeden Speichern und beim Laden von CMSD Daten eine Validierung der Daten gegen das RELAX NG Schema (vgl. Abschnitt 2.3.2) erfolgt, zusätzlich wird eine Pflichtfeldprüfung, basierend auf der RequiredElements.xml, durchgeführt, resultierende Meldungen werden dem Nutzer im Statusfeld des Webfrontends (vgl. Abbildung 80, oben) angezeigt.

Bevor auf den zweiten wesentlichen Bereich des Webfrontends, der Simulationsteuerung und Experimentverwaltung, eingegangen werden kann, bietet sich ein Exkurs in den Bereich Administration und Nutzerverwaltung an, welche umfangreiche Auswirkungen auf alle anderen Bereiche hat. Eine wesentliche Komponente der Administration des Webfrontends bzw. des Frameworks die Pflichtfelddefinition wurde bereits thematisiert. Weitere anwendungsweit gültige Einstellungen, wie Ort (IP und Port) und Version des für Tests verwendeten (Plant Simulation) Modellgenerators, der Port über welchen sich weitere Generatoren registrieren können, Verbindungsdaten zu CMSD Datenquellen (z.B. SAP ERP), gültige Währungseinheiten und Prioritätsregeln werden in einer weiteren XML Datei, der Data.xml, zentral gekapselt. Die hier definierbaren Währungseinheiten und Prioritätsregeln werden genutzt, um entsprechende Auswahlfelder in den Webformularen mit Werten zu befüllen. Weitere Einstellungen werden nicht auf Anwendungsebene sondern nutzerbezogen vorgenommen. Die Nutzung des Webfrontends bedarf prinzipiell einer Anmeldung mit einem gültigen Nutzer (Name und Passwort), dies wurde mittels im ASP.Net verfügbaren Basismechanismen[38] realisiert. Ein angemeldeter Nutzer kann jederzeit eindeutig identifiziert werden, dies wird genutzt um in der UserConfig.xml nutzerbezogene Einstellungen auszulesen. Für jeden Nutzer ist in dieser XML Datei das Modellverzeichnis, das Workspace-Verzeichnis sowie Einstellungen bzgl. Kennzahlen, vgl. Abschnitt 3.4.4 hinterlegt. Das Modellverzeichnis stellt dabei den bereits angesprochenen serverseitigen Speicherort der CMSD Dateien dar, hierbei können ein und derselbe Speicherort für mehrerer Nutzer hinterlegt werden. Dies ermöglicht ein Arbeiten auf gleichen Datenbeständen wobei aber keine Transaktionssicherheit gewährleistet wird.

Der Bereich der Simulationsteuerung und Experimentverwaltung erlaubt dem Nutzer des Webfrontends die Modellgenerierung, einen Simulationslauf bzw. ein komplettes Experiment, d.h. eine Reihe von Läufen zu starten. Im Prototyp des Webfrontends sind

[38] ASP.NET Authentication; vgl. [MSDN2013b]

zwei technisch abweichende Ansätze implementiert. Der erste Ansatz dient vor allem dem Test eines Modells bzw. der Abarbeitung eines einzelnen Simulationslaufs, der zweite Ansatz ermöglicht webbasierte Verteilung von Simulationsläufen/Experimenten, wobei das Ziel ist, zur Absicherung der statistischen Auswertung mehrere unabhängige Läufe auf Basis eines CMSD Modells parallel in mehreren Simulatorinstanzen durchzuführen (vgl. Abbildung 64).

Ansatz eins ermöglicht aktuell allein die Verwendung des Plant Simulation Modellgenerators, in der in den Administrationsdateien eingestellten Version auf dem ebenfalls hinterlegten Host. Die Abarbeitung erfolgt in drei aufeinander aufbauenden Schritten. Zunächst muss ein auf dem Server gespeichertes Modell ausgewählt werden, welches im Folgenden entsprechend generiert sowie initialisiert werden kann (vgl. Abbildung 80, Simulationsmanagement). Hierzu wird die HTTP Schnittstelle des Plant Simulation Modellgenerators genutzt (vgl. Abschnitt 3.4.2.1), über welche vor allem der Pfad des CMSD Files übermittelt wird. Ist die Generierung erfolgreich, wird dem Anwender ein Screenshot des aktuellen Modells wie in Plant Simulation generiert angezeigt. Im zweiten Schritt kann das gewünschte Startdatum sowie die Dauer des Simulationslaufs angegeben werden, wobei der eigentliche Simulationslauf, wiederum über die HTTP Schnittstelle angewiesen wird. Ist der Lauf ohne Fehler beendet, wird dem Nutzer ein Ergebnisbericht präsentiert. Anschließend im dritten und letzten Schritt kann der CMSD Ergebnisexport des Modellgenerators gestartet werden. Die interpretationskonform um Ereignisse und aktuelle Zustände erweiterte CMSD Datei kann abschließend im Webfrontend geladen werden.

Ansatz zwei erlaubt die Verwendung mehrerer Modellgeneratorinstanzen auf ggf. unterschiedlichen Rechnern (vgl. Abbildung 64). Ziel ist es, die Dauer eines Experiments zu senken und die Ressourcenauslastung (Simulatorlizenzen und Rechner) zu verbessern. Grundlegend anzunehmen ist, dass durch Replikation und ggf. mehrere zeitgleiche Webfrontendnutzer eine Vielzahl von Simulationsläufen in einem Zeitraum durchzuführen sind. Hierzu werden vom Nutzer neben den Angaben zur CMSD Datei, Start und Dauer des Experiments zusätzlich die Zahl der gewünschten Simulationsläufe sowie der Name des Experiments abgefragt (vgl. Abbildung 80, Experimentmanagement). Weiterhin ist anzunehmen, dass die verfügbare Simulatorenzahl begrenzt ist, es wird daher unterstellt, dass zwar mehrere Simulatoren verfügbar sind, dass deren Zahl aber kleiner als die Anzahl der aktuell gewünschten Simulationsläufe ist. Dies bedingt schließlich einen Mechanismus zur Verteilung einzelner Simulationsläufe auf die vorhandenen Ressourcen. Prinzipiell wird für die Verteilung vorausgesetzt, dass zum einen sämtlicher Datenaustausch CMSD basiert erfolgt und zum anderen, dass jede Simulatorinstanz sich beim Webfrontend entsprechend registriert hat [BSS2012]. Der Registrationsmechanismus wurde so simpel

wie möglich realisiert und basiert auf einem im Webfrontend integrierten TCP/IP Server, welcher Anmeldungen von Clients entgegennimmt und die Verbindungsdaten speichert. Die einzelnen Simulatorinstanzen werden durch einen Client gesteuert, über welchen alle nötigen Einstellungen gepflegt werden können (vgl. Abbildung 82). Dieser kann bei Bedarf, ebenfalls über TCP/IP, CMSD Datenströme empfangen (CMSD Konzeptmodell) und versenden (Ergebnis als CMSD). Mittels dieses Mechanismus ist es möglich verschiedene Modellgeneratorinstanzen, welche auch ggf. verschiedene Simulatoren nutzen (z.B. Plant Simulation oder SLX) einheitlich anzusprechen. Durch den Registrierungsmechanismus ist ebenfalls eine hohe Flexibilität gegeben, da jederzeit Simulatorinstanzen ausscheiden oder hinzukommen können. Meldet sich eine Instanz nicht ordnungsgemäß ab, wird dies durch periodisch durch den Server versendete Statusnachrichten erkannt.

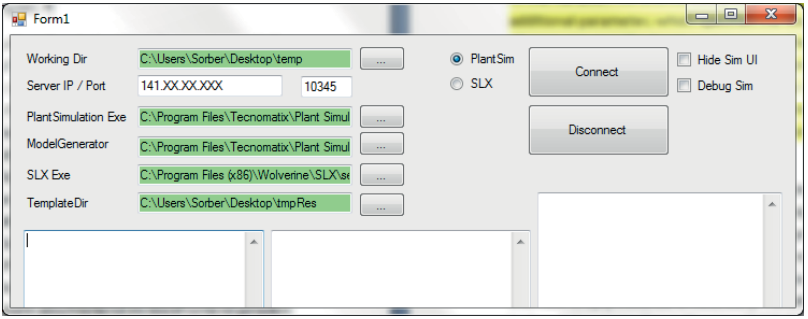


Abbildung 82: Client zur Steuerung der verteilten Modellgeneratoren

Im Detail sind auf Seite des Webfrontends verschiedene Verteilungsmechanismen denkbar. Im Rahmen der Entwicklung des Webfrontends wurden drei Verfahren untersucht (vgl. Abbildung 83).

In Variante A1 und A2 erhält jede Instanz pro Experiment einmalig einen CMSD Datenstrom, sowie ergänzende Informationen bzgl. der Anzahl (rc) von gewünschten Simulationsläufen. In Variante A1 werden alle für ein Experiment definierten Simulationsläufe auf alle zum Start des Experiment verfügbaren Instanzen gleichverteilt aufgeteilt. In Variante A2 erfolgt die Verteilung anhand von in einer Datenbank gespeicherten Performancewerten der einzelnen Instanzen, somit werden schnellere Instanzen mit mehr Läufen beauftragt. In allen Varianten erfolgt am Ende jeden Laufes die Rückmeldung der Ergebnisse in Form eines CMSD Datenstroms. Generell benötigen beide Varianten A1 und A2 zusätzliche Logik zur Verwaltung der Läufe auf Clientseite, zudem sind beide Varianten tendenziell unflexibel bzgl. Änderungen im Clientnetzwerk oder Fehlern einzelner Instanzen, z.B. können während eines Laufes hinzukommende

Instanzen nicht berücksichtigt werden. Zusätzlich ist zu bemerken, dass in Variante A1 die Gesamtperformance nur der der langsamsten Instanz entspricht. Variante A2 weist eine höhere Performance auf, welche aber mit Aufwand zur Ermittlung der Performancekennziffern erkauft wird. Die Ermittlung der Kennziffern zur Performance soll hier nicht näher vertieft werden, kann sich aber als sehr schwierig erweisen, zudem die Performance einzelner Instanzen über die Zeit schwanken kann, z.B. wenn weitere Aufgaben durch einen Rechner parallel wahrgenommen werden [BSS2012, S.9].

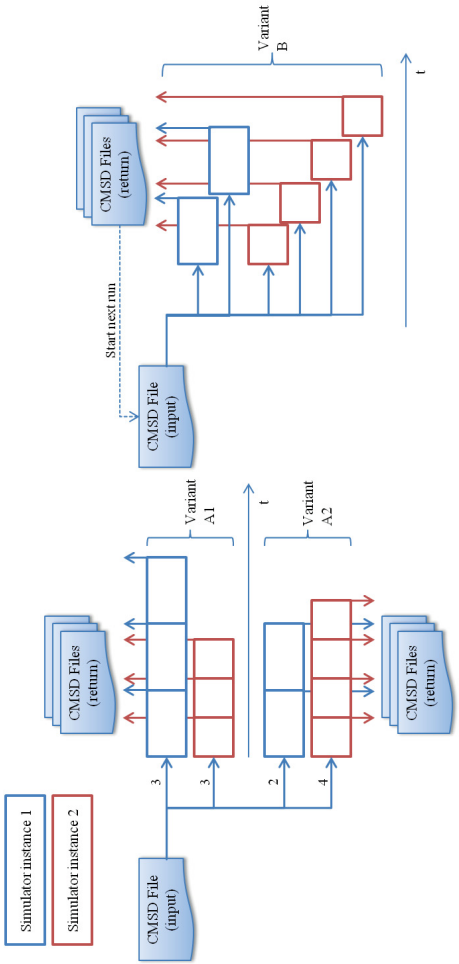


Abbildung 83: Varianten für CMSD basierte Verteilung von Simulationsläufen [BSS2012, S.9]

Variante B ermöglicht, dass alle Steuerungslogik im Webfrontend implementiert wird, da nur einzelne Läufe durch die Instanzen durchgeführt werden müssen. Jeder der Läufe wird separat gestartet sobald Bedarf besteht und entsprechend eine Instanz frei ist. Eine Instanz wird immer als frei gekennzeichnet, wenn sie mit keinem Simulationslauf beauftragt ist, d.h. kein CMSD File übergeben wurde bzw. das Ergebnis bereits vorliegt. Der Bedarf an Simulationsläufen wird auf Serverseite mittels des Auswertens der Liste offener Simulationsläufe realisiert. Dazu wird serverseitig eine anwendungsweit gültige Liste gepflegt, welche alle nötigen Informationen enthält, hierzu zählen u.a.:

- der Pfad zu CMSD Daten, welche simuliert werden sollen,
- der Nutzer für welchen die Ergebnisse verfügbar sein sollen (legt den Speicherort der Ergebnisdaten anhand der spezifischen Nutzerdaten fest),
- der Name des Experiments , unter welchem die Ergebnisse angeordnet werden.

Da Variante B einige entscheidende Vorteile und wenig Nachteile aufweist, wurde aktuell im Webfrontend nur diese implementiert. Nachteil der Varianten ist das höhere Volumen an auszutauschenden CMSD Daten, wobei in heutigen Rechnernetzen die Bandbreite für diese Variante nicht den Flaschenhals darstellen sollte. Vorteile sind der geringere Implementierungsaufwand im Client, dass keinerlei Performancekennziffern benötigt werden, die Flexibilität bzgl. der Anzahl der Clients und der Robustheit bzgl. hinzukommender Simulationsläufe sowie die hohe Gesamtperformance, auf die einzelne langsame Instanzen keinen großen Einfluss haben [BSS2012, S.10f].

Erweiterungsmöglichkeiten des Verteilungsmechanismus sind vielfältig. So kann z.B. an Mechanismen zur Behandlung von Fehlern, z.B. eine beauftragte Simulationsinstanz stürzt ab, oder zur faireren Verteilung der Ressourcen im Multinutzerbetrieb[39] gearbeitet werden.

3.4.4 Simulationsergebnisrepräsentation und Darstellung - WebStatMonitor

Die dritte Komponente des Frameworks ermöglicht es Auswertungen einzelner Simulationsläufe oder ganzer Experimente durchzuführen, indem Kennzahlen (vgl. Abschnitt 2.1.7) und geeignete Diagramme auf CMSD Basis bereitgestellt werden. Hierzu wird die in Abschnitt 3.3.9 beschriebene Abbildung von Ereignis- und Zustandsdaten in CMSD genutzt. Diese Komponente schließt den Kreis, da nun alle Phasen einer Simulationsstudie durch Komponenten des Frameworks unterstützt werden. Um die bereits angerissene webbasierte Simulation zu komplettieren, ist auch diese Komponente als ASP.Net Webanwendung implementiert [BSS2012, Ho2012].

[39] z.Z. FIFO, d.h. ein Nutzer kann mit einen umfangreichen Experiment kann alle anderen Nutzerexperimente blockieren.

Der WebStatMonitor ist vollständig mit dem Webfronend kompatibel und nutzt die gleichen administrativen Möglichkeiten sowie dieselbe Nutzerverwaltung. Für den WebStatMonitor werden zusätzliche Daten nutzerbezogen gespeichert, z.B. Einstellung zu angezeigten Kennzahlen oder benutzerdefinierte Kennzahlen, auf welche im Folgenden an geeigneter Stelle vertiefend eingegangen wird.

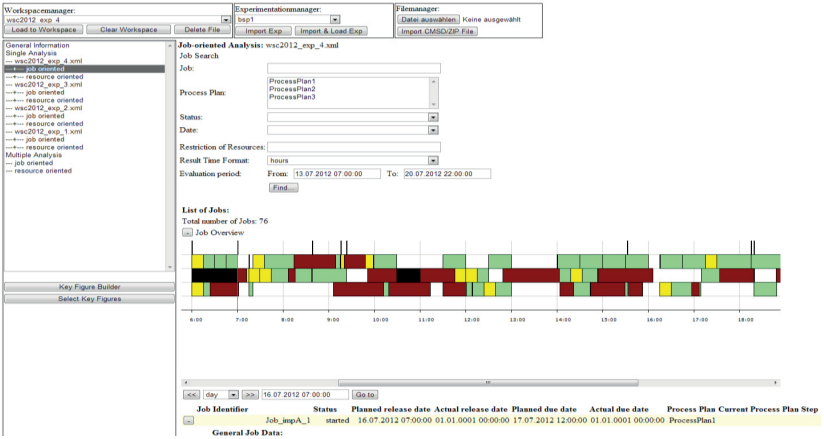
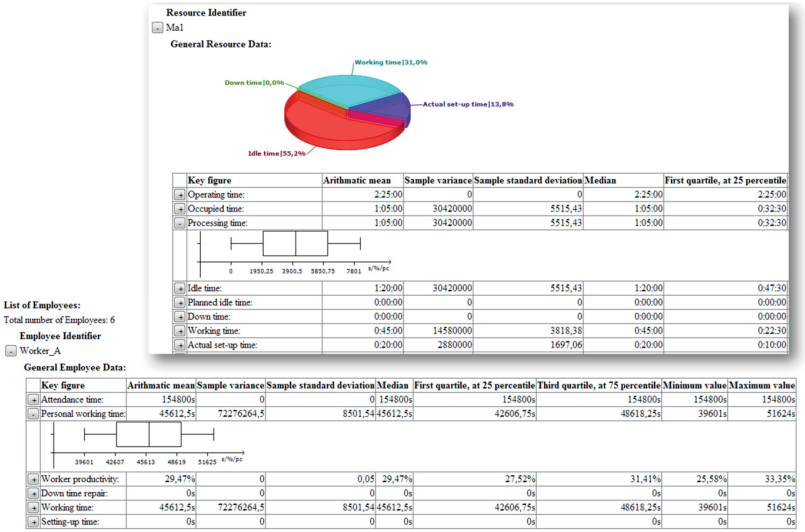


Abbildung 84: Screenshot des WebStatMonitors; Gantt-Diagramm eines Simualtionslaufes

Prinzipiell lassen sich bei der Auswertung zwischen der Auswertung einer CMSD Ergebnisdatei, welche einen einzelnen Simulationslauf repräsentiert (Single Analysis; Einzelanalyse) und der Auswertung über mehrere Dateien hinweg (Multi Analysis; Gruppenanalyse) unterscheiden. Um Auswertungen jeglicher Art durchführen zu können, müssen entsprechend valide CMSD Daten zunächst im WebStatMonitor importiert werden, d.h. es werden serverseitige Kopien der CMSD Daten in entsprechenden nutzerbezogenen Ordnern angelegt. Diese importierten Daten können jederzeit in den aktuellen Workespace (Arbeitsbereich) geladen werden. Der Workspace stellt die Basis aller Auswertungen dar, im Zuge des Ladens einer CMSD Datei werden aus Performancegründen bereits eine Reihe von Basiskennzahlen für diese berechnet. Für jede geladene CMSD Ergebnisdatei kann nun eine Einzelanalyse durchgeführt werden, des Weiteren kann ab zwei geladenen Dateien eine Gruppenanalyse des gesamten Workspaces erfolgen (vgl. Abbildung 84).

Sowohl für Einzel- als auch für Gruppenanalysen kann eine ressourcenorientierte und eine auftragsorientierte Sichtweise unterschieden werden. Die Wahl einer Sichtweise und ob eine Einzel -oder Gruppenanalyse ausgewählt wurde, bestimmt die angezeigten Kennzahlen bzw. deren weitere Aufbereitung und Darstellung sowie die Diagramme, welche zur Verfügung gestellt werden (vgl. Abbildung 84 und Abbildung 85). So sind in

Einzelanalysen verschiedene Gantt-Diagramme verfügbar, in Gruppenanalyse sind diese nicht sinnvoll. In Gruppenanalysen werden jeweils diverse statistische Werte auf einzelnen Kennzahlen gebildet, z.B. Mittelwert, Standardabweichung usw. sowie unterstützend Box-Whisker-Plots[40] eingesetzt.



Key figure	Arithmatic mean	Sample variance	Sample standard deviation	Median	First quartile, at 25 percentile
Operating time:	2:25:00	0	0	2:25:00	2:25:00
Occupied time:	1:05:00	30420000	5515,43	1:05:00	0:32:30
Processing time:	1:05:00	30420000	5515,43	1:05:00	0:32:30
Idle time:	1:20:00	30420000	5515,43	1:20:00	0:47:30
Planned idle time:	0:00:00	0	0	0:00:00	0:00:00
Down time:	0:00:00	0	0	0:00:00	0:00:00
Working time:	0:45:00	14580000	3818,38	0:45:00	0:22:30
Actual set-up time:	0:20:00	2880000	1697,06	0:20:00	0:10:00

Key figure	Arithmatic mean	Sample variance	Sample standard deviation	Median	First quartile, at 25 percentile	Third quartile, at 75 percentile	Minimum value	Maximum value
Attendance time:	154800s	0	0	154800s	154800s	154800s	154800s	154800s
Personal working time:	45612,5s	72276264,5	8501,54	45612,5s	42606,75s	48618,25s	39601s	51624s
Worker productivity:	29,47%	0	0,05	29,47%	27,52%	31,41%	25,58%	33,35%
Down time repair:	0s	0	0	0s	0s	0s	0s	0s
Working time:	45612,5s	72276264,5	8501,54	45612,5s	42606,75s	48618,25s	39601s	51624s
Setting-up time:	0s	0	0	0s	0s	0s	0s	0s

Abbildung 85: Beispiele für Multi Dateien Auswertung im WebStatMonitor; Bearbeitungsstation (oben) und Werker (unten)

In allen Ansichten des WebStatMonitors ist die Zeiteinheit (Sekunden, Minute, Stunde, Tag) variabel einstellbar, alle angezeigten Werte werden sofort entsprechend umgerechnet. Des Weiteren sind je nach Ansicht entsprechende Filtermöglichkeiten vorgesehen, z.B. nach Zeiträumen, Produkten, Bezeichnern usw. Schließlich ist der gesamte WebStatMonitor umfangreich auf die Bedürfnisse des Nutzers einstellbar. So sind 11 auftragsorientierte sowie 40 ressourcenorientierte Kennziffern vorimplementiert. Um die Übersichtlichkeit zu wahren, kann deren Sichtbarkeit jederzeit nutzerbezogen aktiviert bzw. deaktiviert werden, dies ist über ein eigenes Webformular (Select Key Figures) realisiert, die Einstellungen werden in der UserConfig.xml Datei gespeichert. Des Weiteren können eigene Kennziffern bzw. deren Berechnungsvorschriften auf Basis vorhandener Kennziffern bzw. von Basiswerten mittels eines Webformulars (Key Figure Builder) definiert werden (vgl. Abbildung 86).

[40] Box-Whisker-Plots oder Kastengrafiken dienen der grafischen Darstellung der Verteilung kardinalskalierter Daten wobei verschiedene robuste Streuungs- und Lagemaße zusammengefasst werden, meist Median, die zwei Quartile und die beiden Extremwerte [WIKI2013j].

Auch die benutzerdefinierten Kennziffern werden nutzerbezogen persistiert und können dann analog den Basiskennziffern aktiviert bzw. deaktiviert werden.

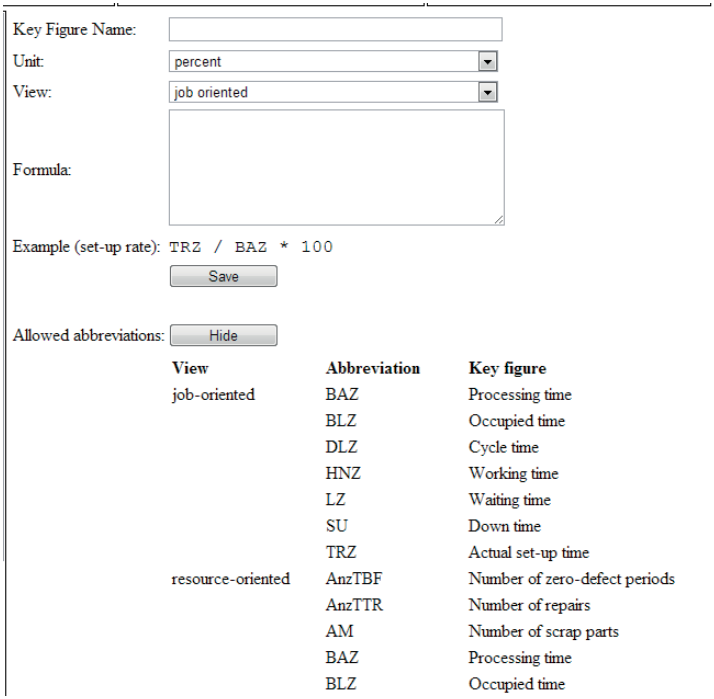


Abbildung 86: Formular zur Definition benutzerdefinierter Kennziffern; Key Figur Builder

Die umfangreichen Möglichkeiten des WebStatMonitors einfach und schnell ein breites Spektrum an Kennziffern zu berechnen und darzustellen, ermöglichen neben dem Einsatz zur Bewertung der Simulationsergebnisse auch den Einsatz im Zuge der Validierung der Modelle indem z.B. Vergleiche mit realen Kennziffern stark vereinfacht werden. Auch denkbar ist es, reale BDE-Daten als CMSD Daten anzulegen und zu Vergleichszwecken zu importieren.

Weitere Details zur Implementierung können [BSS2012] und [Ho2012, S. 52ff] entnommen werden.

Auch der WebStatMonitor kann in Zukunft durch zusätzliche Funktionen erweitert werden, so sind Möglichkeiten zum Vergleich verschiedener Experimente, z.B. unterschiedlicher Systemkonfigurationen, aktuell nicht direkt vorgesehen.

3.5 Fazit

Zusammenfassend kann gesagt werden, dass das Framework aus Vorgehensmodell, CMSD Interpretation und den vorgestellten Softwarekomponenten, insbesondere den Modellgeneratoren, es erlaubt auch nicht triviale Simulationsstudien komplett durchzuführen. Hierbei ist durch die Implementierung aller Nutzeroberflächen als webbasierte Anwendungen eine webbasierte Simulation möglich, ebenfalls ist aber auch eine Integration der Simulation in andere Anwendungen auf Basis des CMSD Standards denkbar, indem z.B. direkt aus einer Anwendung heraus CMSD Daten generiert, die Modellgenerierung und -initialisierung angestoßen sowie abschließend die CMSD Ergebnisdaten ausgewertet werden. In jedem Fall bleibt für den Nutzer bzw. die aufrufende Anwendung der Simulator transparent, d.h. der tatsächlich genutzte Simulator bzw. die genutzten Simulatoren bedingen keine Änderungen an anderen Komponenten. So ist eine Nutzung verschiedener Simulatoren bzw. der Austausch eines Simulators jederzeit möglich, sofern für diesen ein CMSD Modellgenerator implementiert wurde. Die Wahl des Implementierungsansatzes (intern vs. extern) ist hier fallweise zu treffen, wobei für bausteinorientierte Simulatoren eher der interne Ansatz, für Simulationssprachen der externe Ansatz empfohlen wird.

In der Implementierung von CMSD Modellgeneratoren aber auch in der Implementierung von unidirektionalen CMSD Schnittstellen in typischen betrieblichen Anwendungssystemen ist auch eine Aufgabe zu sehen, welche den Erfolg bzw. die Verbreitung des CMSD Standards und des hier vorgestellten Konzepts maßgeblich beeinflusst. Des Weiteren ist es nötig, die CMSD Interpretation und somit auch den Funktionsumfang der Modellgeneratoren wie im Vorgehensmodell vorgesehen durch weitere Anwendungsfälle zu ergänzen. Die Erweiterbarkeit der Interpretation ermöglicht es evolutionär in kleine Schritten neue Anwendungsgebiete zu erschließen.

Schließlich ist zu bemerken, dass die Validierung zwar bereits durch die Möglichkeit, Kennzahlen schnell und mit wenig Aufwand zu berechnen, z.B. im WebStatMonitor, unterstützt wird, aber eine vollautomatische Validierung, die ggf. auch als Auslöser für Adaptionen oder Neugenerierungen der Modelle dient, z.B. in betriebsbegleitenden Anwendungsfällen, derzeit nicht implementiert ist. Die Automatisierung der Validierung stellt einen der aus dieser Arbeit resultierenden spannenden Forschungsbereich dar.

Des Weiteren sind einige große und eine Vielzahl von kleinen Erweiterungen der (Software-) Komponente denkbar. So könnte die Erzeugung und Pflege von CMSD Daten durch den Einsatz von Assistenten (Wizzards)[41] bzw. die Definition von Workflows unterstützt, der Vergleich mehrerer Experimente im WebStatMonitor ermöglicht und die Bildung bzw. die Simulation von Szenarien vereinfacht werden.

[41] "Der Begriff Assistent bezeichnet eine Oberfläche, mittels derer ein Anwender durch mehrere Dialoge für eine ergonomische Dateneingabe [...] geführt wird. Es wird eine Hilfestellung wie beim Ausfüllen von Formularen gegeben" [WIKI2013j].

4 Validierung des Konzeptes/Frameworks

Im Zuge dieses Kapitels soll das gesamte Framework, inklusive der CMSD Interpretation und alle technischen Komponenten bzgl. ihrer Eignung im Kontext der Simulation von Produktionssystemen validiert werden. Hierbei beziehen sich im Folgenden alle Aussagen prinzipiell auf das gesamte Framework, die Validierung einzelner Teile/Komponenten wird nicht separat angestrebt. Zu bemerken ist aber, dass während der Entwicklung der einzelnen Komponenten diese wiederholt strukturiert durchgegangen und gedanklich auf ihre Konsistenz und Konsistenz mit allen anderen Komponenten des Frameworks geprüft wurden. Im Zug der Validierung werden die in Abschnitt 2.2 definierten Anforderungen, soweit nicht bereits abschließend diskutiert, aufgegriffen und durch konkrete Methoden bzw. Tests untersucht.

Die Anforderung Erweiterungsfähigkeit des Ansatzes wurde in den vorherigen Abschnitten, vor allem im Zuge der Betrachtungen des Vorgehensmodells (vgl. Abschnitt 3.1) und der CMSD Interpretation (vgl. Abschnitt 3.3) bereits ausreichend demonstriert. Ebenfalls ist aus allen bisherigen Abschnitten ein hoher Automatisierungsgrad und eine große Flexibilität bzgl. der Datenquellen ableitbar, darüber hinaus werden im Zuge der Validierung ausschließlich die automatisierten Modellgenerierungsmethoden und CMSD als Datenstandard genutzt. Die Anforderungen Flexibilität bzgl. der abzubildenden (Produktions-) Systeme und des Detailierungsgrades sowie die Unterstützung der GoM wurde ebenfalls bereits dargestellt, sollen aber im Folgenden im Fokus der Validierung stehen. So werden verschiedene Testszenarios, welche sich in Größe und Aufbau des abgebildeten Systems unterscheiden, genutzt um der Flexibilität Rechnung zu tragen. Größter Wert wird aber auf Richtigkeit, Klarheit und systematischen Aufbau gelegt, Wirtschaftlichkeit und Vergleichbarkeit wird in Folge der Automatisierung unterstellt.

Die durchgeführten Validierungsmethoden nutzen zwei völlig unterschiedliche Herangehensweisen bzw. wurden unter verschiedenen Bedingungen durchgeführt. Zum einen wurden eine Vielzahl von Laborexperimenten (vgl. Abschnitt 4.1) durchgeführt, welche eine Vielzahl detaillierter Ergebnisse zu einzelnen Simulationsaspekten liefern, zum anderen wurde ein Feldexperiment (vgl. Abschnitt 4.2) unter Praxisbedingungen bei einem KMU durchgeführt. Im Feldexperiment wurden der Natur der Dinge geschuldet weniger bzw. weniger verschiedenartige Experimente durchgeführt, es konnten aber praxisbezogene Erkenntnisse, z.B. bzgl. abzubildender Systemkomponenten oder auch der Handhabbarkeit des Frameworks und der Komponenten für nicht Simulationsexperten gewonnen werden.

Zum Einsatz kamen im Zuge der permanent während der Entwicklung aller Komponenten durchgeführten Validierung verschiedene Testmethoden. So wurden eher subjektive bzw. intuitive Verfahren wie die Animation, strukturiertes Durchgehen ebenso wie objektivere Methoden wie Ereignisvaliditätstest, Trace-Analyse, statistische Techniken bzw. Vergleich mit aufgezeichneten Daten genutzt. Alle genutzten Testverfahren haben spezifische Vor- und Nachteile. So ist z.B. die Animation von Simulationsmodellen ein gerade für erste Tests geeignetes Verfahren, das schnell zu für den Entwickler gut interpretierbaren Aussagen kommt. Für eine umfassende möglichst objektive und gut dokumentierbare Validierung ist Animation dagegen ungeeignet. Aus diesem Grund werden im Folgenden vor allem statistische Verfahren zur Dokumentation der Validierung des Frameworks genutzt, welche ggf. deutlichen Aufwand z.B. zur Datenaufbereitung und -analyse erfordern, welche aber auch zu gut dokumentierbaren, wiederholbaren und weitestgehend objektiven Ergebnissen führen. Allen voran wird der Vergleich von Ereignissen bzw. Kennzahlen (vgl. Abschnitt 2.1.7) und deren statistischen Werten durchgeführt, in einigen Fällen kommen ergänzend Gantt Diagramme zum Einsatz. Alle benötigten Werte und Darstellungen, mit Ausnahme der manuell erstellten Modelle, basieren auf den Funktionen des WebStatMonitors (vgl. Abschnitt 3.4.4).

4.1 Laborexperiment

Unter dem Begriff Laborexperimente werden im Folgenden sämtliche Versuche und deren Auswertungen verstanden, welche nicht mittels Praxisdaten sondern an, meist für bestimmte Aspekte entwickelt, Testdaten/Labordaten vorgenommen wurden. Die Ergebnisse einiger dieser Versuche wurden in wissenschaftlichen Veröffentlichungen zu einzelnen Komponenten des Frameworks bereits angerissen, z.B. in [BS2010a, Fi2010, BSS2011a, BSS2011b, Be2012, BSS2012, Wü2013].

Im Zuge der Validierung des Frameworks wurde eine sehr große Anzahl von Experimenten, vor allem Laborexperimenten, durchgeführt und ausgewertet. So bildet jeder Anwendungsfall (vgl. Abschnitt 3.3) ein Testszenario für die Validierung. Diese Szenarios bilden alle als wesentlich erkannten Anforderungen/Funktionalitäten und deren Einbindung in die Gesamtinterpretation in speziell zugeschnittenen Szenarien ab. Ergänzt werden diese Testszenarios durch komplexere bzw. umfangreichere Testszenarios, welche die Validierung der Skalierbarkeit des Frameworks ermöglichen. In diesen Testszenarios werden keine neuartigen Anforderungen aufgenommen, sondern allein die Zahl der Entitäten, also der Bearbeitungsstationen, Werker, Produkte und Arbeitspläne sowie Aufträge schrittweise erhöht. So wurden Szenarios gebildet mit 5, 10 und 15 Bearbeitungsstationen, mit 12, 18 und 24 Werkern, mit 5, 10, 15 und 20 Produkten (je mit eigenen Arbeitsplänen) sowie mit 100, 200 und 300 Aufträgen in verschiedenen Kombinationen. Des Weiteren kam ein bestehendes und im Rahmen

verschiedener studentischer Arbeiten genutztes Szenario, welches eine Werkstattfertigung detailliert abbildet (vgl. [Hö2010, S. 58-68]; Abbildung 94) sowie ein auf SAP ERP Daten beruhendes Szenario zum Einsatz.

Eine Diskussion aller Validierungsexperimente ist nicht sinnvoll und auch nicht möglich, im Folgenden sollen einige wenige repräsentative Validierungsexperimente beispielhaft vorgestellt werden. Bei allen Experimenten dienen wenn möglich (kleinere, deterministische Systeme) mathematisch ermittelte Lösungen als Referenz, in allen anderen Fällen werden manuell erstellte Simulationsmodelle (in Plant Simulation) als Referenz genutzt, eine als ausreichend anzusehende Zahl an Replikationen wird stets durchgeführt. Des Weiteren wird um die Korrektheit und Vergleichbarkeit der Ergebnisse unabhängig des Modellgenerierungsansatzes und Simulators (intern und extern, vgl. Abschnitt 3.4.2) zu validieren, jedes Experiment für jeden der Generatoren durchgeführt und ausgewertet. Des Weiteren werden für einige Szenarios zusätzlich die Experimente unter parallelem Einsatz beider Generatoren durchgeführt, so werden bspw. bei 10 gewünschten Replikationen jeweils 5 Läufe mit dem internen Plant Simulation Modellgenerator und 5 mit dem externen SLX Modellgenerator durchgeführt, die Ergebnisse aller Läufe werden gemeinsam ausgewertet.

Für rein deterministische Systeme kann die Annahme getroffen werden, dass wenn alle Ereignisse zweier oder mehrerer simulierter oder auch realer Produktionssysteme zu gleichen Zeitpunkten auftreten, diese Systeme für die betrachteten Parameter gleich sind bzw. sich ausreichend gleich verhalten. In solchen Fällen stimmen die Werte der Kennzahlen genau überein, somit gleicht sich auch die Gantt Darstellung vollständig. Eine Replikation ist für diese Fälle nicht nötig.

Ein einfaches rein deterministisches Modell stellt Anwendungsfall 1 (vgl. Abschnitt 3.3.1), hier nur mit 3 Aufträgen, dar. Abbildung 87, Abbildung 88 und Tabelle 40 kann entnommen werden, dass sowohl die Nutzung des SLX Modellgenerators als auch die Nutzung von Plant Simulation als Simulator zu gleichen Simulationsergebnissen führt, zudem decken sich die Ergebnisse mit den Erwartungen bzw. den vorab errechneten Werten.

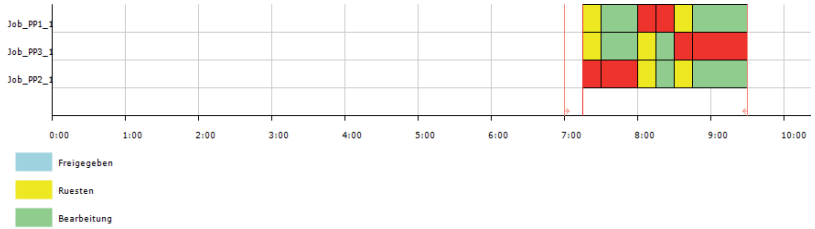


Abbildung 87: SLX Simulationsergebnisse des Anwendungsfalls 1; Bearbeitungssequenz und -dauer der Aufträge als Gantt Diagramm im WebStatMonitor (Auftrag 3 rot markiert)

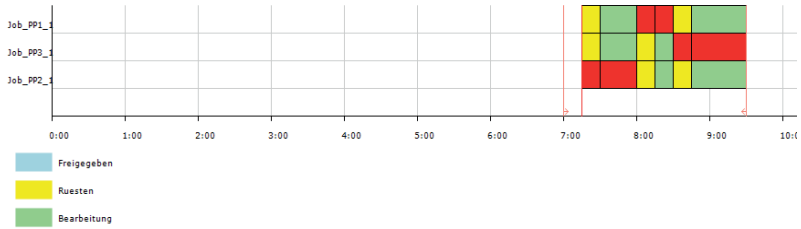


Abbildung 88: Plant Simulation Simulationsergebnisse des Anwendungsfalls 1; Bearbeitungssequenz und -dauer der Aufträge als Gantt Diagramm im WebStatMonitor (Auftrag 3 rot markiert)

Tabelle 40: Vergleich einiger ausgewählter Kennzahlen für Anwendungsfall 1

Kennzahl	*SLX MG*	*Plant Simulation MG*
Mittlere Durchlaufzeit Produkt A	2:25:00 h	2:25:00 h
Mittlere Bearbeitungszeit Produkt A	1:30:00 h	1:30:00 h
Mittlere Arbeitszeit Bearbeitungsstation A	1:30:00 h	1:30:00 h

Ein weiteres deterministisches Beispiel zeigt eine Unterspezifikation im Rahmen der Implementierung der Werker, welche aber nur in wenigen eher theoretischen Konstellationen einen selbst dann zu vernachlässigenden Einfluss hat. Zur Demonstration wird Anwendungsfall 3 (vgl. Abschnitt 3.3.3) ohne Stochastik (Teil von Anwendungsfall 2, Abschnitt 3.3.2) genutzt. In Abbildung 89 und Abbildung 90 ist zu erkennen, dass die Gantt Darstellung im Detail abweicht, gleichzeitig ist der Einfluss auf die Kennwerte gering (vgl. Tabelle 41). Die Kennzahlen entsprechen wiederum den vorab errechneten.

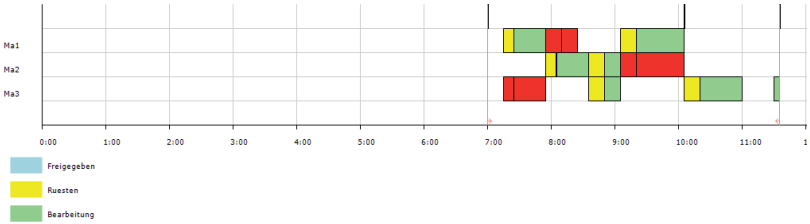


Abbildung 89: SLX Simulationsergebnisse des Anwendungsfalls 3 ohne Stochastik; Bearbeitungssequenz und -dauer der Aufträge als Gantt Diagramm im WebStatMonitor (Auftrag 3 rot markiert)

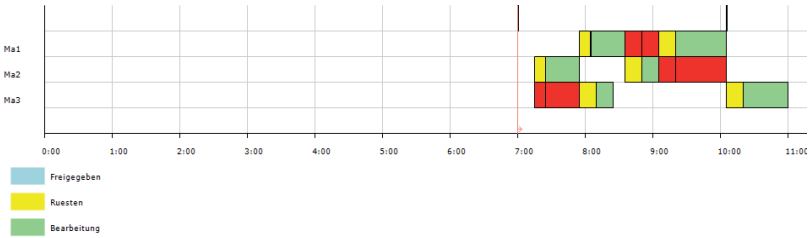


Abbildung 90: Plant Simulation Simulationsergebnisse des Anwendungsfalls 3 ohne Stochastik; Bearbeitungssequenz und -dauer der Aufträge als Gantt Diagramm im WebStatMonitor (Auftrag 3 rot markiert)

Dieses Phänomen beruht auf verschiedenen Mechanismen bei der Abarbeitung von zeitgleich auftretenden Werkeranforderungen in Plant Simulation und SLX. Dies führt, in Fällen in denen ein Werker für verschiede Arbeitsgänge gleichzeitig benötigt würde, zu unterschiedlichen Zuordnungen von Werkern auf Arbeitsgänge. In diesem Bespiel wird ein von mehreren Arbeitsgängen/Stationen benötigter Werker zum Zeitpunkt 7:15 Uhr von SLX der ersten Station und von Plant Simulation der zweiten Station zugeordnet, die jeweils andere muss folglich pausieren, da keine weiteren Werker mit entsprechenden Fähigkeiten zu diesem Zeitpunkt verfügbar sind.

Tabelle 41: Vergleich einiger ausgewählter Kennzahlen für Anwendungsfall 3 ohne stochastische Einflüsse

Kennzahl	*SLX MG*	*Plant Simulation MG*
Mittlere Durchlaufzeit Produkt A	4:35:00 h	4:35:00 h
Mittlere Bearbeitungszeit Produkt A	1:30:00 h	1:30:00 h
Mittlere Arbeitszeit Bearbeitungsstation A	1:30:00 h	1:30:00 h
Arbeitszeit Werker A	1:15:00 h	1:15:00 h
Maximal in Puffer 1 wartende Aufträge	1	2
Maximal in Puffer 2 wartende Aufträge	2	1

Ist ein absolut gleiches Verhalten gewünscht bzw. zur Erreichung der Simulationsziele nötig, ist eine Standardisierung der Zuordnung oder das Beschreiben des Zuordnungsmechanismus z.B. analog der Reihenfolgestrategien möglich. Nach Meinung des Autors ist der Mehraufwand aktuell aber nicht gerechtfertigt, da das Auftreten gleicher Anforderungen zu gleicher Zeit in der Praxis eher unwahrscheinlich ist und ggf. eher eine fehlerhafte Modellierung der Fähigkeiten vorliegt.

In stochastischen Anwendungsfällen ist zwingend eine ausreichende Replikation erforderlich, zudem müssen die Kennzahlenvergleiche umfänglicher gestaltet sein, da nicht auf Gleichheit (z.B. identische Durchlaufzeiten) sondern auf ausreichende Ähnlichkeit (z.B. ähnliche statistische Merkmale der Durchlaufzeit) geprüft werden muss. Hierbei kann jederzeit ergänzend eine Analyse des Konfidenzintervalls zur Anwendung kommen.

Als Beispiel für ein stochastisches Modell wird hier Anwendungsfall 4 (vgl. Abschnitt 3.3.4) genutzt. In Abbildung 91 ist der Ausschnitt eines Gantt Diagramms eines einzelnen Laufes zu sehen. Zu erkennen ist, dass das gewählte Beispiel bereits eine wesentlich höhere Komplexität aufweist als die vorrangegangenen.

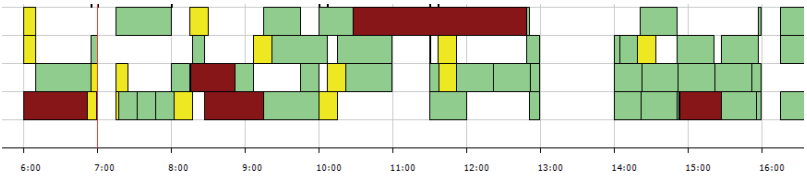


Abbildung 91: Auszug eines Simulationslaufergebnisses des Anwendungsfalls 4; als Gantt Diagramm im WebStatMonitor

Insgesamt wurden für die folgenden Auswertungen 4 Experimentreihen durchgeführt, eine mit einem manuell erstellten Modell (Fall A) sowie drei unter Nutzung des Frameworks, wobei einmal nur Plant Simulation (Fall B), einmal nur SLX (Fall C) und einmal Plant Simulation und SLX (Fall D) als Simulator genutzt wurden. Je automatischer Experimentreihe wurden je 100 Replikationsläufe durchgeführt, welche je 8 Tage simulieren. In Tabelle 42 sind beispielhaft einige ausgewählte Kennziffern und einige der typischerweise erhobenen statistischen Parameter angegeben, ergänzend sind ebenso nur beispielhaft in Abbildung 92 und Abbildung 93 Box Whisker Diagramme für einige dieser abgebildet.

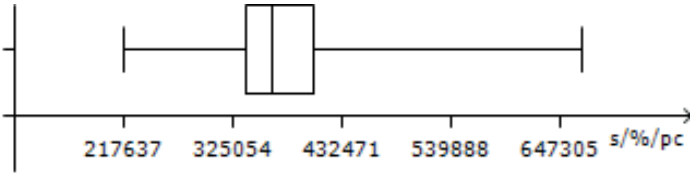


Abbildung 92: Beispiel Box Whisker Diagramm der Kennziffer: Mittlere Durchlaufzeit des Auftrags A1; im Fall B

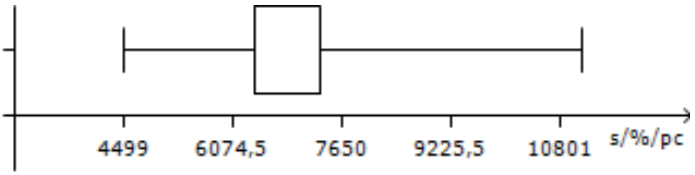


Abbildung 93: Beispiel Box Whisker Diagramm der Kennziffer: Mittlere Rüstzeit der Bearbeitungsstation A; im Fall B

Tabelle 42: Vergleich ausgewählter Kennzahlen des Anwendungsfalls 4

Kennzahl	*Fall*	*Mittelwert*	*Standard-abweichung*	*Min*	*Max*
Mittlere Durchlauf-zeit des Auftrags A1	A	110:44:23 h	32:50:12 h	61:07:03 h	179:28:54 h
	B	111:13:37 h	32:55:49 h	60:27:17 h	179:48:24 h
	C	111:43:17 h	32:56:18 h	60:29:45 h	176:12:48 h
	D	111:38:22 h	33:01:22 h	60:12:17 h	181:33:37 h
***Mittlere Rüstzeit* des Auftrags B1**	A	0:03: 41 h	0:07:50 h	0 h	0:30:00 h
	B	0:03: 36 h	0:08:17 h	0 h	0:30:00 h
	C	0:03: 33 h	0:08:01 h	0 h	0:30:00 h
	D	0:03: 38 h	0:08:22 h	0 h	0:30:00 h

Mittlere Arbeitszeit Bearbeitungsstation A	A	10:56:45 h	1:22:33 h	6:00:00	14:00:00 h
	B	11:03:58 h	1:21:58 h	6:00:00	14:00:00 h
	C	11:12:23 h	1:22:01 h	6:00:00	14:00:00 h
	D	11:05:57 h	1:21:38 h	6:00:00	14:00:00 h
***Mittlere Rüstzeit der* Bearbeitungsstation A**	A	1:50:23 h	0:15:20 h	1:15:00 h	3:00:00 h
	B	1:46:30 h	0:14:50 h	1:15:00 h	3:00:00 h
	C	1:45:45 h	0:14:30 h	1:15:00 h	3:00:00 h
	D	1:43:30 h	0:14:26 h	1:15:00 h	3:00:00 h
Maximal in Puffer 1 (Eingangspuffer A) wartende Aufträge	A	12,44 Stk.	2,01 Stk.	12	18
	B	13,01 Stk.	2,21 Stk.	11	21
	C	13,23 Stk.	2,26 Stk.	11	20
	D	12,55 Stk.	2,44 Stk.	10	21

Den Abbildungen und der Tabelle ist zu entnehmen, dass die Abweichung zwischen den Fällen eher zu vernachlässigen ist. Das Verhalten der automatisch generierten Modelle und hierbei unabhängig des Simulators kann für das Szenario als ausreichend gleich angenommen werden.

Die Betrachtungen zu den Laborexperimenten sollen zwei komplexere Szenarios abschließen, welche im Rahmen der Validierung herangezogen wurden. Zum einen das bereits angesprochene Modell nach Höber (vgl. [Hö2010, S. 58-68) sowie ein Modell, welches auf Basis eines in SAP ERP-System abgebildeten Systems mittels einer CMSD Schnittstelle (vgl. Abschnitt) erzeugt wird [Wü2012].

Das Modell nach Höber lag als Plant Simulation Modell bereits vor (vgl. Abbildung 94), dieses soll als Referenzlösung genutzt werden. Alle Informationen wurden anschließend mittels Webfrontend in CMSD modelliert, wodurch eine automatische Generierung ermöglicht wurde, vgl. Abbildung 95. Bereits der rein optische Vergleich der beiden Modelle (vgl. Abbildung 94 und Abbildung 95) zeigt, dass nur minimale Unterschiede im Layout bestehen. Die Kennzahlenvergleiche zeigen zudem, dass auch das Verhalten der beiden Modelle ausreichend ähnlich ist, somit ist auch für dieses Beispiel von einer validen automatischen Modellgenerierung auszugehen.

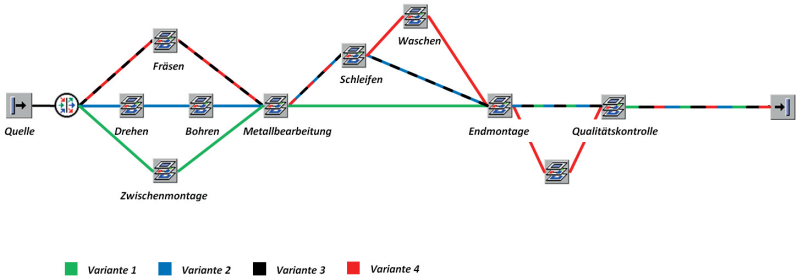


Abbildung 94: Validierungsszenario nach Höber manuell erstelltes Plant Simulation Modell [Hö2011, S. 65]

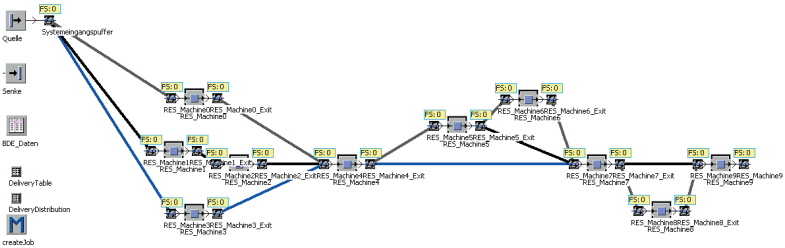


Abbildung 95 Validierungsszenario nach Höber erstellt mittels automatischer CMSD basierter MG in Plant Simulation

Das letzte näher betrachtete Laborexperiment weicht von den anderen bisher betrachteten Experimenten dahingehend ab, dass alle Daten mittels einer im Rahmen einer studentischen Arbeit [Wü2012] entwickelten Schnittstelle direkt aus einem SAP ERP-System stammen, für Daten die nicht explizit in SAP ERP vorlagen, wurden Defaultwerte angenommen, welche aber jederzeit mittels des Webfrontend anpassbar sind. In SAP ERP wurden verschiedene Szenarios modelliert, welche alle eine Werkstattfertigung mit 8-13 teils parallelen Bearbeitungsstationen und bis zu 13 Werkern, mehreren verschiedenen Arbeitsplänen usw. darstellen. Beispielhaft wird im Folgenden eine rein deterministische Variante mit 13 Bearbeitungsstationen, wobei Gruppe 7 aus zwei und Gruppe 8 aus drei parallelen Stationen besteht, und 10 Aufträgen mit einer Losgröße zwischen 5 und 20 sowie vier verschiedenen Arbeitsplänen extrahiert und simuliert, wobei wiederum SLX und Plant Simulation zum Einsatz kommen. Die Auswertung beider Varianten kommt zu denselben Kennzahlen (vgl. Tabelle 43), Abbildung 96 zeigt das entsprechende Gantt Diagramm. Die simulativ

ermittelten Werte entsprechen den vorab berechneten. Die in SLX und Plant Simulation abweichende Werkerzuordnung hat keine Auswirkungen auf die Ergebnisdaten.

Tabelle 43: Vergleich weniger ausgewählter Kennzahlen automatisch generierter Modelle auf Basis von SAP ERP Daten

Kennzahl	*SLX MG*	*Plant Simulation MG*
Mittlere Durchlaufzeit des Auftrags 1 (Job_000001005100)	25:08:00 h	25:08:00 h
Mittlere Arbeitszeit Bearbeitungsstation A (Machine_0008_SIMASCH1_1)	10:44:00 h (53%)	10:44:00 h (53%)
Maximaler Pufferstand Eingangspuffer Bearbeitungsstation A	7	7
Arbeitszeit Werker 8 (Worker_0008_SIMASCH6_1)	11:30:00 h (57,17%)	11:30:00 h (57,17%)

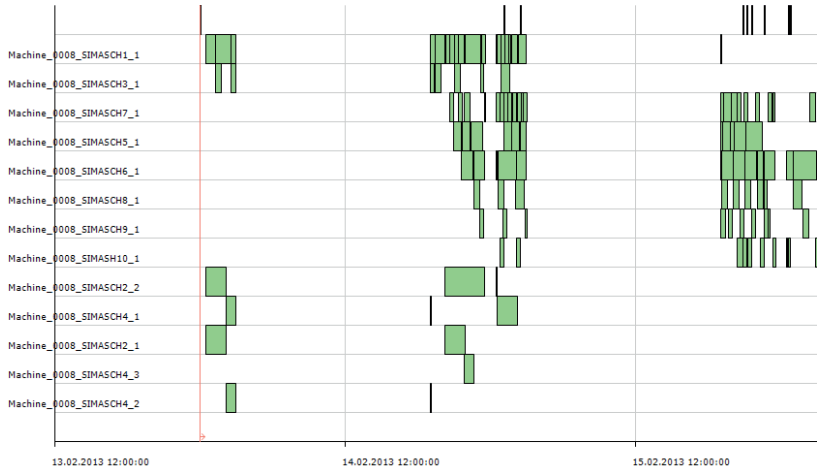


Abbildung 96: Auszug eines Simulationslaufergebnisses basierend auf SAP ERP Daten; als Gantt Diagramm im WebStatMonitor

Die hier aufgezeigten sowie alle weiteren Experimente zeigen, dass eine CMSD basierte automatische Modellgenerierung (vgl. Abschnitt 3.4.2) mit dem in dieser Arbeit vorgestellten Framework erfolgreich simulatorneutral durchgeführt werden kann. Dies wurde für eine Vielzahl, zwar konstruierter aber doch vielgestaltiger und praxisnaher Szenarios nachgewiesen. Auch die Fähigkeit zur Erstellung und Pflege von korrekten und

simulierbaren CMSD Dateien und der ggf. verteilten Ansteuerung der Modellgenerierung mittels des CMSD Webfrontend (vgl. Abschnitt 3.4.3) und die Erzeugung von CMSD Daten mittels Schnittstellen zu Drittsystemen am Beispiel SAP ERP wurde eingehend validiert. Des Weiteren wurden auch die Auswertung von Simulationsexperimenten mittels CMSD und dem WebStatMonitor (vgl. Abschnitt 3.4.4) hinreichend validiert.

4.2 Feldexperiment - Modellgenerierung und Initialisierung von Simulationsmodellen zur Optimierung von Produktionsprozessen eines KMU

Neben den im letzten Abschnitt vorgestellten Laborexperimenten wurde zusätzlich ein Feldexperiment zur Validierung des Frameworks angestrebt, die Ergebnisse des Feldexperiments werden im Folgenden vorgestellt.

Das gesamte Feldexperiment wurde im Rahmen einer studentischen Arbeit [Hä2012] in Kooperation mit einem Partner-KMU durchgeführt, wobei zur Simulation allein Komponenten des Frameworks (vgl. Abschnitt 3.4) zum Einsatz kamen. Konkret wurde vor allem das Webfrontend ergänzt durch entsprechende Schnittstellen sowie der Plant Simulation Modellgenerator genutzt, zur Ergebnispräsentation kam eine sehr frühe Version des WebStatMonitors zum Einsatz. Ziel des Experimentes war es unter anderem zu zeigen, ob und wie CMSD basierte Simulationsmodellgenerierung geeignet ist, in KMUs durch Nicht-Simulationsexperten eingesetzt zu werden, welche weiteren Anforderungen in der Praxis relevant werden und ob das Framework valide Ergebnisse liefert, welche in der Praxis genutzt werden können. Eine rein objektive quantitative Auswertung wie im Zuge der Laborexperimente ist hierbei nicht oder nur schwer möglich, im Folgenden kommen verstärkt qualitative Auswertungen, welche durch Befragungen der Nutzer erhoben wurden, zum Tragen.

Als das potentiell vielversprechendste Anwendungsszenario wurde durch die Geschäftsleitung des KMUs die Unterstützung der operativen Produktionssteuerung durch die betriebsbegleitende Simulation verschiedener Szenarien im Sinne einer einfachen manuellen simulationsbasierten Optimierung identifiziert. Als zweite interessante aber im Rahmen des Feldexperiments nur am Rande betrachtete Anwendung für Simulation wurde die Unterstützung der strategischen Planung und hier vor allem der Absicherung bzgl. Investitionsentscheidungen, z.B. Investitionen in neue Maschinen oder Anlagen durch das KMU genannt. Des Weiteren wurde aus Unternehmenssicht als besonders interessanter Teil des komplexen Fertigungssystems die Vor- und Zwischenfertigung (Zyklus 1 und 2) von Komponenten dargestellt.

Zu bemerken ist, dass die eigentliche Simulation für den Endnutzer vollständig transparent bleiben soll, d.h. während der Tests nutzten die Mitarbeiter des KMU allein das CMSD-Webfrontend, eine CMSD-Schnittstelle zum ERP-System des Unternehmens, welche auch die Variationen der Auftragsparameter erlaubt und somit die Bildung der Optimierungsszenarios ermöglicht, sowie eine ebenfalls webbasierte Ergebnisauswertung. Die Simulation selbst wurde in Plant Simulation über das Webfrontend (vgl. Abschnitt 3.4.3) gesteuert und lief Remote auf Ressourcen des Fachgebiets Wirtschaftsinformatik für Industriebetriebe der TU Ilmenau. Zusätzlich während des Feldexperiments auftretende Anforderungen wurden wie im Vorgehensmodell (vgl. Abschnitt 3.1) vorgesehen, durch den Endnutzer textuell und durch Beispieldateien beschrieben sowie durch den Autor implementiert. Ein Beispiel für eine solche Anforderung ist der Sammelprozess, welcher in Abschnitt 3.3.7 als Teil eines der CMSD Anwendungsfälle aufgenommen wurde.

Die Unternehmensanforderungen sowie die Anforderungen bzgl. der Tests des Frameworks führten zu dem in Abbildung 97 dargestellten Versuchsaufbau. Hierzu wurde zunächst das zu betrachtende Fertigungssystem im Webfrontend als CMSD Konzeptmodell modelliert. Die Modellierung basierte auf einer vorhandenen IST Modellierung der Prozesse des KMU, der Analyse von Daten des ERP-Systems sowie einigen wenigen zusätzlichen manuell erhobenen Daten, z.B. bzgl. Rüstzeit. Diese einmalige Modellierung erwies sich als aufwendig, aber als durchaus mit gegebenen Mitteln realisierbare Aufgabe. Die Nutzung von Schnittstellen bzw. einer Workflowkomponente, welche bei der Erzeugung des Modells unterstützt, wurde als mögliche Verbesserungsoption identifiziert. Die hier nur am Rande betrachteten Versuche bzgl. verschiedener Investitionsalternativen ließen sich ebenfalls problemlos mittels des Webfrontends modellieren.

Zu bemerken ist, dass im Fertigungssystem des KMUs eine Vielzahl von Erzeugnissen produziert wird, für welche verschiedene Fertigungsprozesse/-verfahren möglich sind. So sind meist zwei technologisch unterschiedliche Fertigungsprozesse (L und F) in Zyklus 1 möglich, in Zyklus 2 ist meist sowohl eine hoch automatisierte (A) als auch eine manuelle Bearbeitung (M) möglich. Daraus ergeben sich je Produkt bis zu vier mögliche Arbeitspläne (LM, LA, FM, FA), alle gültigen Arbeitspläne aller aktuell möglichen Produkte sind vollständig im CMSD Konzeptmodell abgebildet. Weitere Analysen der ERP Daten zeigen zudem, dass im abgebildeten System typischerweise hohe Losgrößen bei einer geringen Auftragsvielfalt zu erwarten sind.

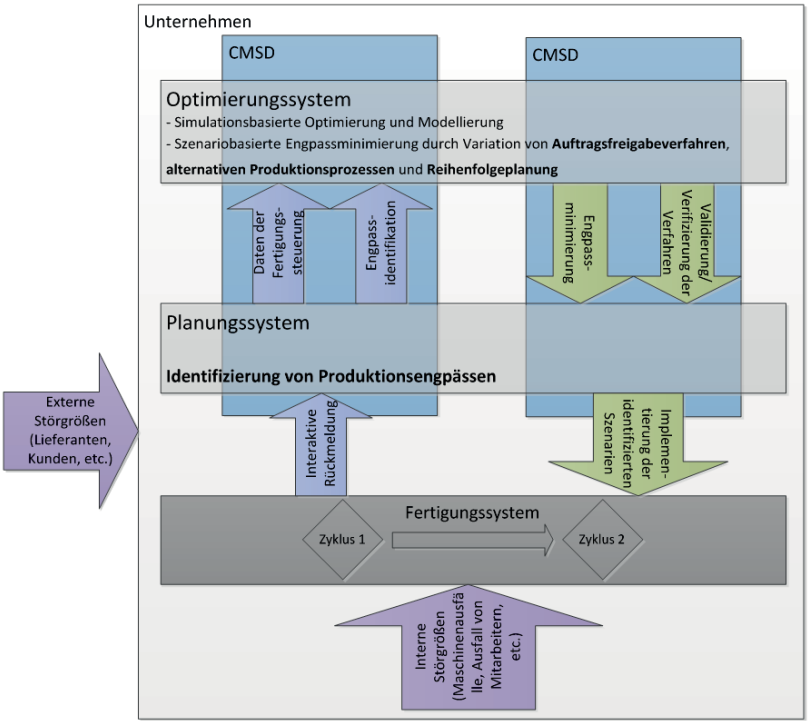


Abbildung 97: Ziel und Zweck der CMSD basierten simulationsbasierten Optimierung und Modellierung (in Anlehnung an [Hä2012, S. 50])

Das erzeugte Modell der Fertigung inklusive aller Ressourcen, Arbeitspläne usw. bildet die Basis aller weiteren Versuche, welche zum Ziel haben einen gegebenen Auftragsmix einer Planungsperiode (zwei Wochen), welcher im ERP-System hinterlegt ist, optimal im Produktionssystem einzulasten. Dazu können verschiedene Parameter der Fertigungsaufträge (Jobs) variiert werden. Die so entstehenden Szenarios werden anschließend simuliert und anhand definierter Systemkennzahlen durch einen Mitarbeiter bewertet. Die Parameterkombination mit der besten Performance wird im Realsystem angewendet. Als für die Steuerung relevante Parameter, d.h. Parameter die sowohl einen signifikanten Einfluss auf die Performance des Systems aufweisen als auch im bestehenden realen Fertigungssystem ohne großen Zusatzaufwand "einstellbar" sind, wurden der Freigabetermin, d.h. der frühste Bearbeitungsbeginn eines Auftrags, die Reihenfolgeregeln in Puffern vor Engpassstationen sowie die Auswahl eines Arbeitsplans aus der Menge möglicher Pläne identifiziert. Die semiautomatische Optimierung ist durch eine überschaubare Anzahl verschiedener Szenarios je

Planungsperiode möglich. Dies resultiert zum einen aus der relativ geringen Zahl der Aufträge je Planungsperiode zum anderen sind je Parameter nur wenige Ausprägungsvarianten, welche im Realsystem möglich sind, vorgesehen, vgl. Tabelle 44 [Hä2012, S.49-53].

Tabelle 44: Parameter zur Szenariobildung im Feldexperiment

Parameter	**Ausprägungen**		
Freigabetermin[42]	Liefertermin - 2 * Bearbeitungszeit	Liefertermin - 2 Wochen	Liefertermin - zufälliger Faktor (1 bis 3) * Bearbeitungszeit
Arbeitspläne	Eine der bis zu vier Arbeitsplanvarianten je Produkt (LM, LA, FM, FA)		
Reihenfolgeregel[43]	FIFO	EDD	SPT

Als zur Bewertung relevante Zielfunktionen werden die Minimierung der durchschnittlichen Durchlaufzeit, der durchschnittlichen Verspätung und die durchschnittliche Terminabweichung durch die Geschäftsleitung des KMU definiert. Alle Szenarios werden ausreichend repliziert, zur Bewertung werden allein Durchschnittswerte der entsprechenden Kennzahlen über alle Läufe genutzt [Hä2012, S.54-56].

Auswertungen der Daten sowie Vergleiche mit Realdaten zeigten, dass die Simulation unter Nutzung des Frameworks zu belastbaren Ergebnissen führt, d.h. dass sich die Simulation und das Realsystem bei gleichen Ausgangsdaten ausreichend gleich verhielten. Somit lassen sich valide Modelle unterstellen. Das Feedback aller projektbeteiligten Mitarbeiter des KMUs war durchweg positiv, wenn auch Verbesserungspotential vor allem in Bezug auf die Usability [44] der Weboberflächen bemerkt wurde. Die Abbildung des betrachteten Fertigungssystems in CMSD war durch Mitarbeiter des KMU, welche keine Simulationsexperten waren, mit wenig Einarbeitungsaufwand möglich. Fehlende Interpretationen konnten wie im Vorgehensmodell vorgesehen leicht ergänzt werden. Als unverzichtbar für den Anwendungsfall der betriebsbegleitenden Simulation zeigten sich die umfangreichen Möglichkeiten zur Initialisierung von Modellen, welche im Framework vorgesehen sind. Auch die Möglichkeiten zur Kennzahlenermittlung überzeugten vollständig.

[42] Der Freigabetermin des Auftrags wird abhängig des geplanten Liefertermins nach verschiedenen Regeln bestimmt, Standardverfahren im KMU ist Liefertermin - 2 Wochen.
[43] Zu Ausprägungen der Reihenfolgeregeln siehe Abschnitt 3.2.1.
[44] Die Usability /Benutzerfreundlichkeit "bezeichnet die vom Nutzer erlebte Nutzungsqualität bei der Interaktion mit einem System" [WIKI2013].

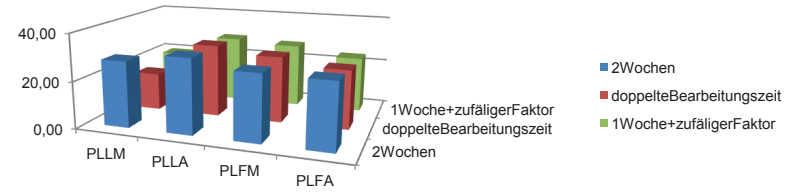


Abbildung 98: Beispielergebnis des Feldexperiments; durchschnittliche Durchlaufzeit (in Tagen) [Hä2012, S.74]

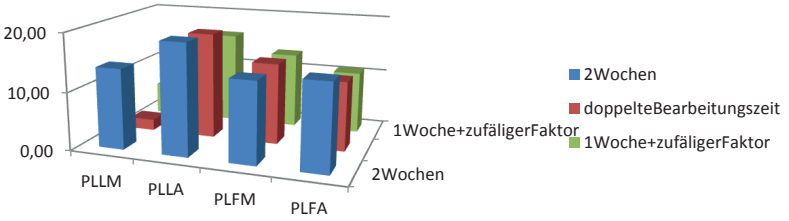


Abbildung 99: Beispielergebnis des Feldexperiments; durchschnittliche Verspätung (Min) [Hä2012, S.74]

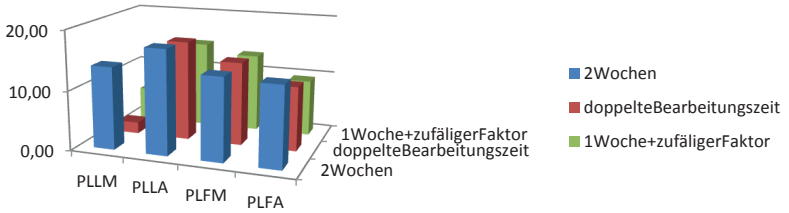


Abbildung 100: Beispielergebnis des Feldexperiments; durchschnittliche Terminabweichung (Min) [Hä2012, S.74]

Abschließend wird nochmals auf die Potentiale der Simulation im Allgemeinen und dem hier vorgestellten Framework im Speziellen für Unternehmen insbesondere KMUs hingewiesen. So kann selbst die einfache manuelle simulationsbasierte Optimierung unter Nutzung des Frameworks zur webbasierten Simulation sehr gut zur Unterstützung der Fertigungssteuerung eingesetzt werden. Im Feldexperiment wurde gezeigt, dass die hier gewählten Szenariovarianten teils erhebliche Unterschiede bzgl. der Zielkennzahlen produzieren (vgl. Abbildung 98, Abbildung 99 und Abbildung 100) [Hä2012, S72ff].

5 Zusammenfassung und Ausblick

In dem folgenden Kapitel wird zunächst die Arbeit kurz zusammengefasst, bevor die Ergebnisse kritisch gewürdigt werden sowie ein Ausblick auf weitere offene Forschungsbereiche und Entwicklungsmöglichkeiten gegeben wird.

5.1 Zusammenfassung

Unstrittig ist, dass die Simulation von Produktions- und Logistiksystemen enormes Potential über den gesamten Produktionslebenszyklus und für verschiedene Frage- bzw. Aufgabenstellungen aufweist. Den Potentialen steht ein nicht unerheblicher Aufwand zur Durchführung von Simulationsstudien gegenüber. Ein Ansatzpunkt diese Aufwände zu senken sowie potentiellen Simulationsnutzern den Einstieg zu erleichtern ist die Automatisierung möglichst vieler Phasen einer Simulationsstudie. Darüber hinaus verspricht die Automatisierung zahlreiche weitere positive Effekte, wie die Beschleunigung einzelner Phasen, die Erhöhung der Modellqualität, die Verbesserung der Wiederholbarkeit usw.

Die hier vorgestellte Arbeit greift genau diesen Ansatz auf. Ziel der Arbeit war es, ein umfassendes Framework zur Integration bzw. Automatisierung der Simulation unter Nutzung von Standards zu entwerfen und zu validieren. Die Kernkomponente bildet hierbei die auf in betrieblichen Informationssystemen über das zu simulierende System vorhandenen Daten basierende automatische Modellgenerierung, -initialisierung und -adaption. Ein weiteres Ziel war es, Methoden zur Abbildung dynamischen Verhaltens im Kontext der automatischen Modellgenerierung zu betrachten und in das Framework zu integrieren sowie ein für die Automatisierung geeignetes Vorgehensmodells vorzuschlagen.

Hierzu wurde zunächst in Kapitel 2 der aktuelle Stand der Forschung im Bereich Simulation von Produktionssystemen und insbesondere der automatischen Simulationsmodellgenerierung analysiert und entsprechende Forschungslücken bzw. -fragen identifiziert. Ebenfalls im Zuge der Recherche wurden potentielle Datenstandards analysiert und bewertet. Im Zuge dieser Analyse wurde der Core Manufacturing Simulation Data (CMSD) Standard als am besten geeigneter Standard zum Datenaustausch im Kontext der Simulation von Produktionssystemen identifiziert und im Laufe der Arbeit genutzt.

In Kapitel 3 wurde das entwickelte Konzept bzw. Framework vorgestellt sowie alle Teilkomponenten einzeln beleuchtet. Hierzu wurde zunächst ein für die automatische Modellgenerierung angepasstes Vorgehensmodell vorgestellt, welches die Automatisierung von Simulationsstudien ermöglicht. Ebenfalls sind im angepassten

Vorgehensmodell Schnittstellen zur Entwicklung der benötigten Softwarekomponenten vorgesehen. Anschließend wurde die Abbildung dynamischen Verhaltens, als eine der großen offenen Fragestellungen im Kontext der automatischen Modellgenerierung, thematisiert. Neben der Klassifizierung bestehender Ansätze wurde zunächst ein eigener, auf künstlichen neuronalen Netzen basierender Ansatz vorgestellt, der ermöglicht das Modellverhalten durch Lernprozesse auf Basis von BDE-Daten zu adaptieren. Zentrales Element dieses Kapitels bzw. der gesamten Arbeit ist aber die Erstellung und Validierung einer praxistauglichen Interpretation des CMSD Standards, d.h. das Festlegen der konkreten Nutzung einzelner Elemente des Standards zur Modellgenerierung, -initialisierung und Ergebnisauswertung. Diese Interpretation bildet eine Referenzimplementierung des Standards und erfolgt anhand aufeinander aufbauender Anwendungsfälle, welche im Kontext der Werkstattfertigung angesiedelt sind. Abschließend wurden die im Rahmen der Arbeit entworfenen und implementierten Softwarekomponenten vorgestellt. Hierzu zählen neben den eigentlichen Modellgeneratoren das Webfrontend, welches vor allem zur Verwaltung und Pflege von CMSD Modellen sowie zur Steuerung des Gesamtframeworks, inklusive der Experimentverteilung dient, sowie der WebStatMonitor, der vielfältige Funktionalitäten zur Auswertung von CMSD Ergebnisdaten, einzelner Simulationsläufe oder ganzer Experimente bereitstellt. Für die Modellgeneratoren, welche simulatorspezifisch zu implementieren sind, wurden zwei unterschiedliche Implementierungsansätze vorgestellt für die jeweils ein Prototyp entwickelt wurde. Für den internen Ansatz wurde der weit verbreitete bausteinorientierte Simulator Plant Simulation der Firma Siemens PLM Software genutzt, für den externen Ansatz die Simulationssprache SLX der Wolverine Software Corporation gewählt. Ergänzend werden Schnittstellen zu betrieblichen Anwendungssystemen, z.B. SAP ERP beispielhaft angerissen.

Im Rahmen des 4. Kapitels wurde das gesamte Framework validiert. Hierzu kamen zum einen Laborexperimente zum anderen ein Feldexperiment zum Einsatz. Ziel war es neben den Softwarekomponenten vor allem auch die CMSD Interpretation zu testen. Im Feldexperiment wurde ebenfalls das angepasste Vorgehensmodell praktisch erprobt. Die Ergebnisse der Validierung waren allesamt überaus zufriedenstellend. So zeigte sich, dass das Framework nicht nur in konstruierten akademischen Testfällen, sondern gerade auch in der Praxis eingesetzt werden kann und zu belastbaren Ergebnissen führt.

5.2 Kritische Würdigung

Bezugnehmend auf die in Abschnitt 2.4 dargestellten Forschungsfragen kann festgestellt werden, dass das vorgestellte Gesamtkonzept inklusive des Vorgehensmodells eine Modellierungsmethodik darstellt, welche sinnvoll automatische Modellgenerierung und -adaption ermöglicht. Es ist festzustellen, dass der CMSD Datenstandard zur Abbildung von Konzeptmodellen im Kontext der Simulation von Produktionssystemen geeignet ist, alle für die Simulation relevanten Daten abzubilden. Besonders herauszuheben ist hierbei, dass alle Phasen einer Simulation unterstützt werden. So konnte gezeigt werden, dass gerade auch die Initialisierung von Simulationsmodellen auf Basis von CMSD Daten sowie auch die Speicherung von Ergebnisdaten in CMSD möglich ist. Ebenso wird durch das Framework die Adaption von Modellen auf allen Ebenen unterstützt. Es konnte in der vorliegenden Arbeit gezeigt werden, dass die eingangs des Kapitels angesprochenen positiven Effekte z.B. bzgl. der sinkenden Aufwände, erhöhten Durchführungsgeschwindigkeit einzelner Simulationsstudien, der Bedienbarkeit durch Nicht-Simulationsexperten oder der produktlebenszyklusphasenübergreifenden Nutzbarkeit erreichbar sind. So sind beispielsweise durch die automatische Modellgenerierung und -initialisierung erhebliche Zeiteinsparungen realisierbar, wobei eine gleichbleibend hohe Modellqualität gewährleistet wird. Ebenso werden durch die Nutzung der Schnittstellen aber auch des Webfrontends z.B. Aufwände bei der Datenbeschaffung gesenkt sowie periodische und phasenübergreifende Modellerstellung und -nutzung ermöglicht. Im Rahmen dieser Arbeit und in diversen Veröffentlichungen auf namhaften Konferenzen, z.B. der Winter Simulation Conference wurde demonstriert, dass automatische Modellgenerierung auf Basis von CMSD Daten prinzipiell möglich ist. Angerissen wurden zudem Ansätze wie Validierung mittels automatisch erzeugten Kennzahlen oder auch die Automatisierung des Datenaustauschs. Dies ermöglicht erst die im Konzept angestrebte permanente bzw. zumindest periodische Validierung der Modelle und somit das Erkennen von Abweichungen zwischen Simulationsmodell und dem Realsystem, welche z.B. aus Lerneffekten im Realsystem resultieren. Erkannte Abweichungen können genutzt werden um durch Adaption bzw. Neugenerierung von Simulationsmodellen auch über lange Zeiträume die Qualität der Modelle zu erhalten oder gar zu erhöhen. Hierzu sind weitere Forschungsarbeiten denkbar und sinnvoll (vgl. 5.3 Ausblick).

Ein Ergebnis dieser Arbeit ist, dass die Nutzung des CMSD Standards eine Interpretation des Standards voraussetzt, da dieser im Detail verschiedene Abbildungsmöglichkeiten einzelner Sachverhalte erlaubt. So wurde eine praxistaugliche Interpretation des Standards entwickelt und validiert. Ebenfalls sind im Standard kaum Pflichtfelder definiert, eine Definition dieser ist aber Grundvoraussetzung zur Nutzung des Standards. Die Pflichtfelddefinition stellt somit ebenfalls einen essentiellen Teil der Interpretation dar. Die hier vorgestellte (Referenz-) Interpretation ist eine mögliche

gültige Interpretation, welche es erlaubt viele Aspekte einer Werkstattfertigung abzubilden, erhebt aber nicht den Anspruch auf Vollständigkeit, so ist nicht jedes denkbare Szenario in der Interpretation enthalten. Dies stellt aber keine relevante Einschränkung des Ansatzes dar, da im Zuge der Entwicklung der CMSD Interpretation von Anfang an auf Erweiterbarkeit und Anpassbarkeit bzgl. der abbildbaren Systeme geachtet wurde. Die Möglichkeiten zur Erweiterung des Standards selbst um zusätzliche Elemente, im Form von Properties bzw. Erweiterungen von Aufzählungsdatentypen (Enumerations) wurden möglichst sparsam eingesetzt, um sich nicht unnötig weit vom Basisstandard zu entfernen. Zusammenfassend kann konstatiert werden, dass die hier geleistet Arbeit bzgl. der CMSD Interpretation unabhängig des restlichen Frameworks eine solide Grundlage für weitere Interpretationen und Entwicklungen darstellt. Des Weiteren wurden Vorschläge zur Erweiterung bzw. Modifikation des Standards erarbeitet.

Die entwickelten Softwarekomponenten des Frameworks basieren alle auf der CMSD Interpretation und ermöglichen zum einen im Zusammenspiel das Durchführen kompletter Simulationsstudien, zum anderen können Komponenten ebenso allein genutzt werden. So können z.B. die Modellgeneratoren auch zur direkten Integration in betriebliche Anwendungssysteme eingesetzt oder die CMSD Schnittstelle zum Datenaustausch zwischen betrieblichen Anwendungssystemen wie ERP und MES genutzt werden.

Die Ergebnisse der Validierung zeigen, dass die Modelle auch für die Praxis eine ausreichend gute Qualität aufweisen. Es traten hierbei keine signifikanten Unterschiede zwischen den Implementierungen in unterschiedlichen Simulatoren auf, auch wenn den Implementierungen verschiedene Ansätze (intern vs. extern) zugrundeliegen. Des Weiteren wurde gezeigt, dass das Gesamtframework und das angepasste Vorgehensmodell praxisgeeignet sind. Durch die für die Softwarekomponenten gewählten Implementierungstechniken konnte zudem eine Umgebung geschaffen werden, welche webbasierte Simulation bzw. Simulation as a Service ermöglicht.

Ferner konnte ein Beitrag in dem weiterhin hoch wichtigen Bereich der Abbildung dynamischen Verhaltens im Kontext der Modellgenerierung erbracht werden. So wurde eine Klassifizierung möglicher Ansatze vorgeschlagen sowie ein innovativer Ansatz, welcher auf der Nutzung von künstlichen neuronalen Netzen basiert, vorgestellt.

5.3 Ausblick

In der hier vorliegenden Arbeit wurde ein breites Spektrum an Fragestellungen im Kontext der automatischen Generierung von Simulationsmodellen von Produktionssystemen erfolgreich bearbeitet. Wie aber bereits in den entsprechenden Kapiteln und der Zusammenfassung angedeutet, sind weitere teilweise erst aus dem Ergebnis der Arbeit resultierende Themen nicht abschließend bearbeitet. Hierbei sind sowohl theoretisch wissenschaftliche als auch eher technisch praktische oder organisatorische Themen bzw. Aufgaben gleichermaßen vertreten.

Eine große anstehende Herausforderung ist in der Steigerung der Bekanntheit und Verbreitung des höchst interessanten Core Manufacturing Simulation Data (CMSD) Standards sowie des damit einhergehenden Konzeptes der Modellgenerierung zu sehen. Die hier vorliegende Arbeit sowie die Veröffentlichungen des Autors und einiger weniger weiterer Wissenschaftler zum Thema sind hier als ein erster guter Beitrag zu sehen aber allein nicht ausreichend. Die Kommunikation von Case Study oder Success Storys scheint hier ein besonders geeignetes Hilfsmittel zur Steigerung der Aufmerksamkeit in Praxis und Wissenschaft. Des Weiteren müssen entsprechende Softwarelösungen, welche mehr oder weniger direkt durch interessierte Unternehmen und Forschungseinrichtungen einsetzbar sind entwickelt, vermarktet bzw. bereitgestellt werden. Zu den zu entwickelnden Softwarekomponenten zählen neben weiteren simulatorspezifischen CMSD basierten Modellgeneratoren vor allem auch geeignete Schnittstellen zu betrieblichen Standardsoftwaresystemen.

Allgemein ist weiterhin das hier dargestellte Konzept in der Praxis zu erproben. Neben der Steigerung der Bekanntheit des CMSD Standards und der Verbreitung des Konzepts der CMSD basierten Modellgenerierung ist zudem an der CMSD Interpretation zu arbeiten, d.h. weitere Anwendungsfälle, die die Abbildung weiterer Produktionssysteme ermöglichen oder verbessern müssen in die Gesamtinterpretation eingepflegt werden. Zudem ist die bestehende Interpretation permanent weiter zu testen bzw. zu validieren und ggf. zu verbessern. Bei allen Erweiterungen ist neben der Grundbedingung der korrekten Abbildung der Sachverhalte, vor allem auf Standardkonformität, Kompatibilität mit der bestehenden Interpretation und der Erhaltung der Erweiterbarkeit dieser zu achten. Die Ergebnisse sollten nach Möglichkeit Interessierten frei zugängig gemacht werden, z.B. durch Publikation in Zeitschriften oder auf Fachtagungen. Aus Sicht des Autors erscheint die Betrachtung von Szenarien, welche Fördertechnik wie z.B. Fließbänder, Heber, Stapler usw. beinhalten, aktuell als besonders lohnenswertes Forschungsfeld.

Ein weiterer Ansatzpunkt für Forschungen stellt das im Rahmen dieser Arbeit ausgeklammerte Gebiete des teilautomatischen Experimentdesigns bzw. der

Szenarioanalyse (Generierung und Auswahl) dar, welches gesondert am Fachgebiet Wirtschaftsinformatik für Industriebetriebe der TU Ilmenau wissenschaftlich bearbeitet wird.

Des Weiteren ist wie bereits angedeutet das Problem der Abbildung von dynamischem Verhalten in Simulationsmodellen weiterhin nicht vollständig gelöst, mögliche Forschungsansätze wurden im entsprechenden Abschnitt skizziert.

Ferner sind weitere Anwendungspotentiale des CMSD Standards denkbar. So wurden Möglichkeiten der unternehmensübergreifenden Simulation, z.B. von Supply Chains bisher nicht beleuchtet, die für die CMSD ebenfalls ein potentiell geeigneter Datenstandard sein könnten. Ebenso sind die Möglichkeiten der in CMSD abgebildeten Konzeptmodelle z.B. zur automatischen Modellsegmentierung bisher weitestgehend ungenutzt.

Literaturverzeichnis

[Ac1996] Acél P. P.: *Methode zur Durchführung betrieblicher Simulationen : effiziente Optimierung der diskreten Simulation*. Dissertation, ETH Zürich, 1996.

[Al2012] ALTOVA: *XMLSpy - XML Editor zum Modellieren, Editieren, Transformieren & Debuggen von XML-Technologien*. Abruf am 08.08.2012, http://www.altova.com/de/xmlspy.html.

[ARN2006] Arnold D.: *Intralogistik. Potentiale, Perspektiven, Prognosen*. Springer, Heidelberg, 2006.

[AT2000] Anderl R., Trippner D.: *STEP, Standard for the Exchange of Product Model Data : eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP)*. Teubner, Stuttgart, 2000.

[Ay2008a] Aydt H., Turner S.J., Cai W., Low M.Y.H., Lendermann P., Gan B.P.: *Symbiotic Simulation Control in Semiconductor Manufacturing*. In Bubak M., van Albada G.D., Dongarra J., Sloot P.M.A.(Hrsg.): Proceeding of the 8th International Conference of Computational Science (ICCS 2008). June 23-25, 2008. Kraków, Polen, S.26-35.

[Ay2008b] Aydt H., Turner S.J., Cai W., Low M.Y.H.: *Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology*. In: Proceeeding of the 22nd Workshop on Principles of Advanced and Distributed Simulation (PADS'08). June 3-6, 2008. Roma, Italien, S.109-116.

[Ba2000] Banks J.: *Simulation in the future*. In: Joines J. A., Barton R. R., Kang K., Fishwick P. A. (Hrsg.): Proceedings of the 2000 Winter Simulation Conference. December 10-13, 2000. Orlando, FL,USA, S. 1558-1567.

[Ba2003] Balci O.: *Validation, verification, and certification of modelling and simulation applications*. In: Chick S., Sanchez P.J., Ferrin E., Morrice D.J. (Hrsg) : Proceedings of the 2003 Winter Simulation Conference. December 7-10, 2003. New Orleans, LA, USA, S.150-158.

[Ba2008] Bangsow S.: *Fertigungssimulationen mit Plant Simulation und SimTalk*. Carl Hanser Verlag, München, 2008.

[Ba2010] Barton R. R.: *Simulation experiment design*. In: Johansson B., Jain S., Montoya-Torres J., Hugan J., Yücesan E. (Hrsg.): Proceedings of the 2010 Winter Simulation Conference. December 5-8, 2003. Baltimore, MD, USA, S. 75-86.

[BA2004] Beck K., Andres C.: *Extreme Programming Explained. Embrace Change*. 2.Auflage, Addison Wesley, Boston, 2004.

[BDK2004] Becker J.; Delfmann P.; Knackstedt R.: *Adaption fachkonzeptioneller Referenz-Prozessmodelle*. In: Industriemanagement 1/2004, S.19-22.

[Be1998] Becker J.: *Die Grundsätze ordnungsmäßiger Modellierung und ihre Einbettung in ein Vorgehensmodell zur Erstellung betrieblicher Informationsmodelle.* In: Proceeding der Fachtagung Modellierung betrieblicher Informationssysteme. 15.-16. Oktober, 1998. Koblenz. Abruf am 14.09.2011. http://www.wi-inf.uni-duisburg-essen.de/MobisPortal/pages/rundbrief/pdf/Beck98.pdf.

[Be2001] Beck K., Beedle M., van Bennekum A., CockburnA., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R.C., Mellor S., Schwaber K., Sutherland J., Thomas D.: *Manifest für Agile Softwareentwicklung*. Abruf am 28.01.2013. http://agilemanifesto.org/iso/de/.

[Be2008] Becker T.: *Prozesse in Produktion und Supply Chain optimieren*. Springer Verlag, Berlin, 2008.

[Be2010] Bergmann S.: *Automatic Generation of Adaptive Simulation Models for Production*. In: Johansson B., Jain S., Montoya-Torres J., Hugan J., Yücesan E. (Hrsg.): General Posters of the 2010 Winter Simulation Conference. December 5-8, 2010. Baltimore, MD, USA.

[Be2011] Bergmann S.: *Automatische Generierung adaptiver und lernfähiger Modelle.* In: Eymann T.(Hrsg.):Tagungsband zum Doctoral Consortium der WI 2011, Bayreuther Arbeitspapiere zur Wirtschaftsinformatik. Zürich, 2011, S.9-16.

[Be2012] Bergmann S., Stelzer S., Wüstemann S., Strassburger S.: *Model generation in SLX using CMSD and XML Stylesheet Transformations*. In: Laroque C., Himmelspach J., Pasupathy R., Rose O., Uhrmacher A.M. (Hrsg): Proceddings of the 2012 Winter Simulation Conference. December 9-12, 2012. Berlin, Germany.

[BGB2007] Baumann H., Grässle P., Baumann P.: *UML 2 projektorientiert*. Galileo Computing, 2007.

[Br2010] Breitner M. H.: *Vorgehensmodell*. In: Gronau N., Sinz E., Becker J. Suhl L., Kurbel K. (Hrsg.): Enzyklopädie der Wirtschaftsinformatik. Oldenbourg Wissenschaftsverlag, München, 2010.

[BRA2008] Bracht U., Rooks T., Adrian R.: *Virtuelle Logistikplanung für die Montage im Rahmen der Digitalen Fabrik*. In: Advances in Simulation for Production and Logistics Applications, Stuttgart, Fraunhofer IRB Verlag, 2008, S.439-447.

[BRS1995] Becker J., Rosemann M., Schütte R.: *Grundsätze ordnungsmäßiger Modellierung*. In: Wirtschaftsinformatik, 37 (1995) 5, S. 435–445

[BS2010a] Bergmann S., Strassburger S.: *Generierung und Integration von Simulationsmodellen unter Verwendung des Core Manufacturing Simulation Data (CMSD) Information Model.* In: Zülch G., Stock P. (Hrsg.): Proceeding der 14. ASIM-Fachtagung Simulation in Produktion und Logistik - Integrationsaspekte der Simulation: Technik, Organisation und Personal. Karlsruhe, 2010, S. 461-468.

[BS2010b] Bergmann S., Strassburger S.: *Challenges for the automatic generation of simulation models for production systems.* In Proceedings of the 2010 Summer Simulation Multiconference (SummerSim'10). 11-15. July 2010. Ottawa, Canada, S.545-549.

[BS2011] Bergmann S., Stelzer S.: *Approximation of dispatching rules in manufacturing control using artificial neural networks.* In: Proceeding of the 25th IEEE/ACM/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2011). 15-17. Juni 2011. Nice, France, S. 94-101.

[BSS2011a] Bergmann S., Stelzer S., Strassburger S.: *Initialization of Simulation Models using CMSD.* In: Jain S., Creasey R.R., Himmelspach J., White K.P., Fu M. (Hrsg): Proceedings of the 2011 Winter Simulation Conference. December 11-14, 2011. Phoenix, AZ, USA, S. 2228 - 2239.

[BSS2011b] Bergmann S., Stelzer S., Strassburger S.: *Automatische Generierung und Initialisierung von Simulationsmodellen unter Verwendung des Core Manufacturing Simulation Data (CMSD) Information Model.* In: Müller E., Spanner-Ulmer B. (Hrsg.): Proceedings der 14. Tagen des Betriebs- und Systemingenieurs (TBI'11). Wissenschaftliche Schriftenreihe des IBF. 23.-24. November 2011. Chemnitz, S. 323-332.

[BSS2012] Bergmann S., Stelzer S., Strassburger S.: *A new web based method for distribution of simulation experiments based on the CMSD standard*. In: Laroque C., Himmelspach J., Pasupathy R., Rose O., Uhrmacher A.M. (Hrsg): Proceddings of the 2012 Winter Simulation Conference. December 9-12, 2012. Berlin, Germany.

[BSS2013] Bergmann S., Stelzer S., Strassburger S.: *On the use of artificial neural networks in simulation based manufacturing control*. In: Fowler J, Lee L. H., Taylor S. J. E.: JOS Journal of Simulation, April 2013, Abruf am 30.04.2013, http://www.palgrave-journals.com/jos/journal/vaop/ncurrent/pdf/jos20136a.pdf.

[BV2008] Bankhofer U., Vogel J.: *Datenanalyse und Statistik: Eine Einführung für Ökonomen im Bachelor*. Gabler, Wiesbaden, 2008.

[BWG2009] Bracht U., Wenzel S., Geckler D.: *Digitale Fabrik: Methoden und Praxisbeispiele*. Springer Verlag, Berlin 2009.

[Ca2002] Carson J.S.: *Model verification and validation*. In: Yücesan E., Chen C.H., Snowdon J.L. Charnes J.M. (Hrsg): Proceedings of the 2002 Winter Simulation Conference. December 8-11, 2002. San Diego, CA, USA, S. 52-58.

[Cy1989] Cybenko G.: *Approximation by superpositions of a sigmoidal function*. In: Mathematics of Control, Signals, and Systems (MCSS), 1989 Volume 2, S.303–314.

[DG2010] Doberenz W., Gewinnus T.: *Visual C# 2010 – Grundlagen und Profiwissen*. Hanser Verlag, München, 2010.

[Dr2010] Drath R. (Hrsg.): *Datenaustausch in der Anlagenplanung mit AutomationML - Integration von CAEX, PLCopen XML und COLLADA*. Springer, Heidelberg, 2010.

[Ec2002] Eckardt F.: *Ein Beitrag zu Theorie und Praxis datengetriebener Modellgeneratoren zur Simulation von Produktionssystemen*. Aachen: Shaker, 2002.

[Ev1996] Eversheim W.: *Produktionstechnik und -verfahren*. In: Kern W., Schröder H.-H., Weber, J. (Hrsg.): Handwörterbuch der Produktionswirtschaft. 2. Auflage, Schäffer-Poeschel, Stuttgart, 1996.

[FC2010] Fonseca i Casas, Pau: *Using Specification and Description Language to define and implement discrete simulation models*. In: Proceedings of the 2010 Summer Simulation Multiconference (SummerSim'10). 11-15. July 2010. Ottawa, Canada, S. 419- 426

[FFS2011] Fandel G., Fistek A., Stütz S.: *Produktionsmanagement (2. Auflage)*. Springer Verlag, Heidelberg, 2011.

[Fo2011] Foskett S.: *Defining Failure: What Is MTTR, MTTF, and MTBF?.* Abruf am 13.09.2012, http://blog.fosketts.net/2011/07/06/defining-failure-mttr-mttf-mtbf/.

[Fr2003] Franke C.: *Feature-basierte Prozesskettenplanung in der Montage als Basis für die Integration von Simulationswerkzeugen in der Digitalen Fabrik.* Bley H., Hirt G., Weber C. (Hrsg.) : Schriftenreihe Produktionstechnik - Band 28, Universität des Saarlandes Produktionstechnik, Saarbrücken, 2003.

[FR2004] Fowler J.W., Rose O.: *Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems*. In: Simulation, San Diego, CA, 80(2004)9, S. 469-476.

[Ga2013] Gabler Verlag, Gabler Wirtschaftslexikon: *Produktionsprozessplanung*. Abruf am 04.03.2013, http://wirtschaftslexikon.gabler.de/Archiv/57176/produktionsprozessplanung-v5.html.

[GD2013] Garcia A. A., Drath R.: *AutomationML™ verbindet Werkzeuge der Fertigungsplanung Hintergründe und Ziele*. Abruf am 14.03.2013 https://www.automationml.org/o.red/uploads/dateien/1314344567-automationml_whitepaper.pdf.

[Gy2008] Gyimesi M.: *Web Services with generic simulation models for discrete event simulation*. In: Mathematics and Computers in Simulation,2008 Volume 79, S. 964-971.

[He1997] Henriksen J. O.: *The Power and Performance of Proof Animation*. In: Andradóttir S., Healy K. J., Withers D. H., Nelson B. L. (Hrsg.): Proceedings of the 1997 Winter Simulation Conference. December 7-10, 1999. Atlanta, GA, USA, S. 574-580.

[He1999] Henriksen J. O: *SLX - The X is for eXtensibility.* In: Farrington P. A., Nembhard H. B., Sturrock D. T., Evans G. W. (Hrsg): Proceedings of the 1999 Winter Simulation Conference. December 5-8, 1999. Phoenix, AZ, USA, S. 167-175.

[He2001] Hetem V: *Integrating capacity simulation into process planning.* In: Peters B.A., Smith J.S., Medeiros D.J., Rohrer M.W. (Hrsg.): Proceedings of the 2001 Winter Simulation Conference. December 9-12, 2001. Arlington, VA, USA, S. 1470 -1472.

[HMK2011] Hedler M., Montero Pineda M., Kutscherauer N.: *Schematron: Effiziente Business Rules für XML-Dokumente*. Dpunkt Verlag ,Heidelberg, 2011.

[Ho2007] Hotz I.: *Ein Simulationsbasiertes Frühwarnsystem zur Unterstützung der operativen Produktionssteuerung und -planung in der Automobilindustrie*. Dissertation, Otto von Guericke Universität, Magdeburg, 2007. Abruf am 02.06.2010. http://diglib.uni-magdeburg.de/Dissertationen/2007/inghotz.pdf

[Ho2009] Vonhoegen H.: *Einstieg in XML - Grundlagen, Praxis, Referenz; 5. Auflage*. Galileo Press, Bonn, 2009.

[Hö2010] Höber, C.: *Exceptionsbasiertes MES als Regelungsansatz für Produktions- und Logistiksysteme*. Diplomarbeit, TU Ilmenau, Ilmenau, 2010.

[HRM2007] Huang E., Ramamurthy R., McGinnis L. F.: *System and simulation modeling using SysML*. In: Henderson S.G., Biller B., Hsieh M.-H., Shortle J., Tew J.D., Barton, R.R. (Hrsg.): Proceedings of the 2007 Winter Simulation Conference. December 9-12, 2007. Washington, D.C., USA, S. 796-803.

[HST2005] Hanisch A., Schulze T., Tolujew J.: *Initialization of online simulation models*. In: Kuhl M. E., Steiger N. M., Armstrong F. B., Joines J. A. (Hrsg.): Proceedings of the 2005 Winter Simulation Conference. December 4-7, 2005, Orlando, FL, USA, S.1795-1803.

[HSV2011] Huang Y., Seck M.D., Verbraeck A.: *From data to simulation models: component-based model generation with a data-driven approach*. In Jain S., Creasey R.R., Himmelspach J., White K.P., Fu M. (Hrsg): Proceedings of the 2011 Winter Simulation Conference. December 11-14, 2011. Phoenix, USA, S. 3724- 3734 .

[Hu2006] Huang S.Y., Cai W., Turner S.J., Hsu W.J., Zhou S., Low M.Y.H., Fujimoto R., Ayani R.: *A Generic Symbiotic Simulation Framework*. In: Proceeding of the20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06). May 16-19, 2006. Singapore, S. 131.

[ISO22400-2] International Organization for Standardization (ISO): *Automation systems and integration - Key performance indicators (KPIs) for manufacturing operations management - Part 2: Definitions and descriptions (22400-2).*

[ISO10303-1] International Organization for Standardization (ISO): *Industrial automation systems and integration Product data representation and exchange - Part 1: Overview and fundamental principles (ISO 10303-1)*.

[ISO19757-2] International Organization for Standardization (ISO): *Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG (ISO/IEC 19757-2:2008)*.

[ISO19757-3] International Organization for Standardization (ISO): I*nformation technology -- Document Schema Definition Languages (DSDL) -- Part 3: Rule-based validation -- Schematron (ISO/IEC 19757-3:2006)*.

[ITU Z.100] Telecommunication standardization sector of the International Telecommunication Union: *Specification and description language (SDL) - ITU-T Recommendation Z.100*. Abruf am 05.06.2012 http://www.itu.int/ITU-T/studygroups/com10/languages/Z.100_1199.pdf.

[Jo2007] Johansson M., Johansson B., Skoogh A., Leong S., Riddick F., Lee Y.T., Shao G., Klingstam P.: *A test implementation of the core manufacturing simulation data specification.* In: Henderson S.G., Biller B., Hsieh M.-H., Shortle J., Tew J.D., Barton, R.R. (Hrsg.): Proceedings of the 2007 Winter Simulation Conference. December 9-12, 2007. Washington, D.C., USA, S. 1673-1681.

[KG1995] Kosturiak J., Gregor M.: *Simulation von Produktionssystemen*. Springer, Berlin, 1995.

[Kl2013] Kloos O.: *Generierung von Simulationsmodellen auf der Grundlage von Prozessmodellen - Ein Transformationsmodell Ansatz*. Ilmenau: Dissertation, TU Ilmenau, 2013.

[KN2010] Kloos O., Nissen V.: *Vom Prozess zur Simulation - ein Transformationsmodell-Ansatz*. In: Claus T., Herrmann F. (Hrsg.): Proceedings des Workshops „Simulation als betriebliche Entscheidungshilfe“, 14. ASIM-Fachtagung „Simulation in Produktion und Logistik“, Karlsruhe, 2010, S. 105-119.

[Kr2013] Kriesel D.: *Ein kleiner Überblick über Neuronale Netze*. Abruf am 07.02.2013. http://www.dkriesel.com/_media/science/neuronalenetze-de-zeta2-2col-dkrieselcom.pdf.

[Ku2006] Kühn W.: *Digitale Fabrik: Fabriksimulation für Produktionsplaner*. Hanser Verlag, München, 2006.

[La2006] Lacy L. W.: Interchanging Discrete-Event Simulation Process-Interaction Models using the Web Ontology Language – OWL. Orlando, Dissertation, Department of Industrial Engineering, University of Central Florida.

[La2007] Law A. M.: *Simulation Modeling and Analysis, 4th Edition*. McGraw-Hill, Boston, 2007.

[La2008] Law A. M.: *How to build valid and credible simulation models.* In: Mason S. J., Hill R. R., Mönch L., Rose O., Jefferson T., Fowler J.-W. (Hrsg.): Proceedings of the 2008 Winter Simulation Conference. December 7-10, 2008. Miami, FL, USA, S. 39-47.

[LK2000] Law A. M., Kelton W. D.: *Simulation Modeling and Analysis*. McGraw-Hill, Boston, 2000.

[LC1991] Law A.M., McComas M.G.: *Secrets of successful simulation studies*. In: Nelson B.L., Kelton W.D., Clark G.M. (Hrsg.): Proceedings of the 1991 Winter Simulation Conference. December 8-11, 1991. Phoenix, AZ, USA, S. 21-27.

[Le2008] Leong S. K., Johansson M., Johansson B., Lee, Y. T., Riddick, F. H.: *A Real World Pilot implementation of the Core Manufacturing Simulation Information Model*. In: Proceedings of the 2008 Summer Simulation Multiconference (SummerSim'10). 16-19 June 2008. Edinburg, Scotland, UK.

[LLR2006] Leong S., Lee Y.T., Riddick F.: *A Core Manufacturing Simulation Data Information Model for Manufacturing Applications*. In: Proceedings of the 2006 Fall Simulation Interoperability Workshop. Simulation Interoperability Standards Organization. Orlando: IEEE, 2006.

[LS1995] Lorenz P., Schulze T.: *Layout Based Model Generation*. In: Alexopoulos C., Kang K., Lilegdon W.R., Goldsman D.(Hrsg.): Proceedings of the 1995 Winter Simulation Conference. December 3-6, 1995. Arlington, VA, USA, S. 728-735.

[LZ1996] Lee C.H., Zobel R.N.: *Representation of Simulation Model Components for Model Generation and a Model Library*. In: Proceedings of the 29th Annual Simulation Symposium. April 8-11, 1996. New Orleans, LA, USA, S.193-201.

[Mc2003] McLean C., Leong S., Harrell C., Zimmerman P.M., Lu R.F.: *Simulation standards: current status, needs, and future directions*. In: Chick S., Sanchez P.J., Ferrin E., Morrice D.J. (Hrsg) : Proceedings of the 2003 Winter Simulation Conference. December 7-10, 2003. New Orleans, LA, USA, S. 2019-2026.

[Me2005] Mertens P., Bodendorf F., König W., Picot A., Schumann M., Hess T.: *Grundzüge der Wirtschaftsinformatik*. 9.Aufl., Springer, Berlin, 2005.

[Mo2006] Moorthy S: *Advanced 3D Factory Design integrated with Throughput Simulation*. In: Proceedings of the PLM World 2006. May 8-12, 2006, Long Beach, CA, USA.

[Mö2006] Mönch L.: *Agentenbasierte Produktionssteuerung komplexer Produktionssysteme*. Deutscher Universitäts-Verlag, Wiesbaden , 2006.

[Mö2013] Mönch L.: *Betriebsdatenerfassung*. In: Gronau N., Sinz E., Becker J., Suhl L., Kurbel K. (Hrsg.): Enzyklopädie der Wirtschaftsinformatik. Oldenbourg Wissenschaftsverlag, München, 2013. Abruf am 08.02.2013, http://www.oldenbourg.de:8080/wi-enzyklopaedie/lexikon/informations systeme/Sektorspezifische-Anwendungssysteme/Produktionsplanungs--und--steuerungssystem/Manufacturing-Execution-System/Betriebsdaten erfassung.

[MS2010] Müller-Sommer H.: *Aufwände der Belieferungssimulation - eine Umfrage im VDA Arbeitskreis Ablaufsimulation*. Abruf am 23.03.2011. http://www.dbthueringen. de/servlets/DocumentServlet?id=15869.

[MS2012] Meyer T., Straßburger S.: *Using protocol state machines to support simulation-based emulation projects*. In: Proceedings of the European Simulation and Modelling Conference 2012. Essen, S.234-238.

[MS2013] Müller-Sommer H.: *Wirtschaftliche Generierung von Belieferungssimulationen unter Verwendung rechnerunterstützter Plausibilisierungsmethoden für die Bewertung der Eingangsdaten*. Dissertation, Ilmenau, 2013.

[MSC2013] Microsoft Corporation: *ASP.Net*. Abruf am 30.04.2013. http://www.asp.net/.

[MSDN2013a] MSDN: *Übersicht über den ASP.NET-Sitzungszustand*. Abruf am 03.05.2013, http://msdn.microsoft.com/de-de/library/ms178581(v=vs.100).aspx.

[MSDN2013b] MSDN: *ASP.NET Authentication*. Abruf am 03.05.2013, http://msdn.microsoft.com/de-de/library/eeyk640h(v=vs.71).aspx.

[MU2009] McGinnis L. F., Ustun V.: A *simple example of SysML-driven simulation*. In: Rossetti M. D., Hill R. R., Johansson B., Dunkin A., Ingalls R. G. (Hrsg.): Proceedings of the 2009 Winter Simulation Conference. December 13-16, 2009. Austin, TX,USA, S. 2206-2218.

[Ne2007] Nebl T.: *Produktionswirtschaft (6. Auflage)*. Oldenbourg Verlag, München, 2007.

[NRA2008] Nyhuis P., Reinhart G., Abele E.: *Wandlungsfähige Produktionssysteme - Heute die Industrie von morgen gestalten*. PZH Produktionstechnisches Zentrum, Hannover, 2008.

[OMG2012a] Object Management Group: *OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.3*. Abruf am 04.06.2012. http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/

[OMG2012b] Object Management Group: *OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.3*. Abruf am 04.06.2012. http://www.omg.org/spec/ UML/2.3/Superstructure/PDF/.

[OMG2012c] Object Management Group: *Systems Modeling Language (OMG SysML), Version 1.3*. Abruf am 12.06.2012. http://www.omg.org/spec/SysML/1.3/.

[Ös2010] Österle H., Becker J:, Frank u., Hess T., Karagiannis D., Krcmar H., Loos P., Mertens P., Oberweis A., Sinz E. J.: *Memorandum zur gestaltungsorientierten Wirtschaftsinformatik*. In: Zeitschrift für betriebswirtschaftliche Forschung (zfbf) 62, September 2010, S.662-679.

[PF2002] Palmer S.R., Felsing J.M.: *A Practical Guide to the Feature-Driven Development.* Prentice Hall International, Upper Saddle River, 2002.

[Pi2002] Pinedo M.: Scheduling: *Theory, Algorithms, and Systems*. Prentice Hall, Engelwood Cliffs, NJ, 2002.

[Ra2003] Rabe M.: *Modellierung von Layout und Steuerungsregeln für die Materialflusssimulation*. Fraunhofer IPK/IRB Verlag, Berlin, 2003.

[RG2008] Reinhart G., Gyger T.: *Identification of implicit strategies in production control*. In: Proceeding of the 2008 Industrial Engineering and Engineering Management (IEEM 2008). December 8-11, 2008. Singapore, S.302-306.

[RL2008] Riddick F., Lee T.Y.: *Representing layout information in the CMSD specification.* In: Mason S. J., Hill R. R., Mönch L., Rose O., Jefferson T., Fowler J.-W. (Hrsg.): Proceedings of the 2008 Winter Simulation Conference. December 7-10, 2008. Miami, FL, USA, S 1777-1784.

[Ro2004] Robinson S.: *Simulation: the practice of model development and use*. John Wiley & Sons, Chichester, 2004.

[Ro2009] Rooks T.: *Rechnergestützte Simulationsmodellgenerierung zur dynamischen Absicherung der Montagelogistikplanung bei der Fahrzeugneutypplanung im Rahmen der Digitalen Fabrik*. Dissertation. Shaker, Aachen, 2009.

[RSW2008] Rabe M., Spieckermann S., Wenzel S.: *Verifikation und Validierung für die Simulation in Produktion und Logistik - Vorgehensmodelle und Techniken.* Springer Verlag, Heidelberg, 2008.

[Sa1982] Sargent R.G.: *Verification and validation of simulation models*. In Cellier F.E.(Hrsg.) Progressin modeling and simulation. Acadamic Press, London, 1982, S. 159-169.

[Sa2011] Sargent R.G.: *Verification and validation of simulation models*. In: Jain S., Creasey R.R., Himmelspach J., White K.P., Fu M. (Hrsg.): Proceedings of the 2011 Winter Simulation Conference. December 11-14, 2011. Phoenix, USA, S. 183 - 193.

[Sa2013] Saxonica Limited: *SAXON - The XSLT and XQuery Processor. Abruf am 29.04.2013. http://saxon.sourceforge.net/.*

[SBM2010] Strassburger S., Bergmann S., Müller-Sommer H.: *Modellgenerierung im Kontext der Digitalen Fabrik – Stand der Technik und Herausforderungen.* In: Zülch G., Stock P. (Hrsg.): Proceedings der 14. ASIM-Fachtagung Simulation in Produktion : APersonal. Karlsruhe, 2010, S. 35-42.

[SBS2012] Stelzer S., Bergmann S., Strassburger S.: *Generation of alternatives for model predictive control in manufacturing.* In: Affenzeller M., Bruzzone A.G., De Felice F., Del Rio Vilas D., Frydman C., Massei M., Merkuryev Y. (Hrsg.): Proceedings of the International Conference on Modeling and Applied Simulation 2012. September 19-21, 2012. Wien, A, S. 7-16.

[Sc2002] Schulz R.: *Parallele und Verteilte Simulation bei der Steuerung komplexer Produktionssysteme*. Dissertation, Ilmenau, 2002.

[Sc2012] Schwichtenberg H.: *Microsoft ASP.NET 4.0 mit Visual C# 2010*. Microsoft Press Deutschland, Unterschleißheim, 2011.

[Se2005] Selke C.: *Entwicklung von Methoden zur automatischen Simulationsmodellgenerierung*. München: Dissertation, Technische Universität München, 2005.

[SH2002] Schulze T., Henriksen J. O: *Simulation Needs SLX*. Abruf am 26.04.2013, http://isgwww.cs.uni-magdeburg.de/pelo/sa/SimulationNeedsSLX.pdf.

[Si2009] Silver G. A., Bellipady K. R., Miller J. A., Kochutt K. J., York W.: *Supporting interoperability using the Discrete-Event Modeling Ontology (DeMO)*. In: Rossetti M. D., Hill R. R., Johansson B., Dunkin A., Ingalls R. G. (Hrsg.): Proceedings of the 2009 Winter Simulation Conference. December 13-16, 2009. Austin, TX, USA, S. 1399-1410.

[SISO2011] Simulation Interoperability Standards Organization (SISO): Policies & Procedures. Simulation Interoperability Standards Organization Executive Committee. Abruf am 2.05.2011. http://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=31765&PortalId=0&TabId=105.

[SISO2012a] Simulation Interoperability Standards Organization (SISO): *Standard for: Core Manufacturing Simulation Data – UML Model.* Core Manufacturing Simulation Data Product Development Group. Abruf am 14.02.2012. http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=31457.

[SISO2012b] Simulation Interoperability Standards Organization (SISO): *Standard for: Core Manufacturing Simulation Data – XML Representation.* Core Manufacturing Simulation Data Product Development Group Abruf am 14.02.2012. http://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=32733&PortalId=0&TabId=105.

[Si2013] Siemens: *Plant Simulation*. Abruf am 24.04.2013, http://www.plant-simulation.com/de/.

[SM2001] Sly D., Moorthy S.: *Simulation data exchange (SDX) implementation and use*. In: Peters B.A., Smith J.S., Medeiros D.J., Rohrer M.W. (Hrsg.): Proceedings of the 2001 Winter Simulation Conference. December 9-12, 2001. Arlington, VA, USA, S. 1473 -1477.

[Sp1995] Splanemann R.: *Teilautomatische Generierung von Simulationsmodellen aus systemneutral definierten Unternehmensdaten*. Bremen: Dissertation, Universität Bremen, 1995.

[So2000] Son Y.J.: *Automatic generation of simulation models from neutral libraries: An example*. In: Joines J. A., Barton R. R., Kang K., Fishwick P. A. (Hrsg.): Proceedings of the 2000 Winter Simulation Conference. December 10-13, 2000. Orlando, FL,USA, S. 1558-1567.

[Sp2005] Spieckermann S.: *Diskrete, ereignisorientierte Simulation in Produktion und Logistik - Herausforderungen und Trends*. In: Schulze T., Horton G., Preim B., Schlechtweg S. (Hrsg.): Proceedings der 16. Simulation und Visualisierung. März 3-4, 2005, Magdeburg, S.3-14.

[SR2001] Son Y.J., Wysk R.A.: *Automatic simulation model generation for simulation-based, real-time shop floor control*. In: Computers in Industry, 2001 Volume 45, Issue 3, S 291–308.

[SR2009] Schönherr O., Rose O.: *First steps towards a general SysML model for discrete processes in production systems*. In: Rossetti M. D., Hill R. R., Johansson B., Dunkin A., Ingalls R. G. (Hrsg.): Proceedings of the 2009 Winter Simulation Conference. December 13-16, 2009. Austin, TX,USA, S. 1711-1718.

[SR2010] Schönherr O., Rose O.: *Modellierung mit SysML zu Abbildung von Produktionsprozessen*. In: Zülch G., Stock P. (Hrsg.): Proceeding der 14. ASIM-Fachtagung Simulation in Produktion und Logistik - Integrationsaspekte der Simulation: Technik, Organisation und Personal. Karlsruhe, 2010, S.453-459.

[SR2011] Schönherr O., Rose O.: *A General Model Description for Discrete Processes*. In: Jain S., Creasey R.R., Himmelspach J., White K.P., Fu M. (Hrsg): Proceedings of the 2011 Winter Simulation Conference. December 11-14, 2011. Phoenix, AZ, USA, S. 2206-2218.

[SRS2010] Siegwart H., Reinecke S., Sander S.: *Kennzahlen für die Unternehmensführung*. Haupt, Bern, 2010.

[St2006] Straßburger S., Seidel H., Schady R., Masik S.: *Werkzeuge und Trends der digitalen Fabrikplanung - Analyse der Ergebnisse einer Onlinebefragung*. In: Wenzel S. (Hrsg.): Proceedings der 12. ASIM-Fachtagung Simulation in Produktion und Logistik, Kassel, 2006, S.391-402.

[Ta2012] Taylor S. E. J., Fishwick P. A., Fujimoto R. Uhrmacher A.M., Page E.H., Wainer G.: *Panel on Grand Challenges for Modeling and Simulation*. In: Laroque C., Himmelspach J., Pasupathy R., Rose O., Uhrmacher A.M. (Hrsg): Proceddings of the 2012 Winter Simulation Conference. December 9-12, 2012. Berlin, Germany.

[TM2011] Thiers G., McGinnis L. F.: *Logistics systems modeling and simulation*. In: Jain S., Creasey R.R., Himmelspach J., White K.P., Fu M. (Hrsg): Proceedings of the 2011 Winter Simulation Conference. December 11-14, 2011. Phoenix, AZ, USA, S. 1536-1546.

[Va2006] van der Vlist E.: *Relax NG*. O'Reilly Media, Sebastopol, CA, USA, 2006.

[VDI2893] VDI - Verein Deutscher Ingenieure: *VDI Richtlinie 2893 Auswahl und Bildung von Kennzahlen für die Instanthaltung.* . Beuth, Berlin, 2006

[VDI3633-1] VDI - Verein Deutscher Ingenieure: *VDI Richtlinie 3633-1 Simulation von Logistik- Materialfluss- und Produktionssystemen - Blatt 1 Grundlagen*. Beuth, Berlin, 2008.

[VDI3633-3] VDI - Verein Deutscher Ingenieure: *VDI Richtlinie 3633-3 Simulation von Logistik- Materialfluss- und Produktionssystemen - Blatt 3 Experimentplanung und -auswertung*. Beuth, Berlin, 2008.

[VDI4400-2] VDI - Verein Deutscher Ingenieure: *VDI Richtlinie 4400-2 Logistikkennzahlen für die Produktion - Blatt2*. Beuth, Berlin, 2004.

[VDI4499] VDI - Verein Deutscher Ingenieure: *VDI Richtlinie 4499 Digitale Fabrik - Grundlagen.* Beuth, Berlin, 2008.

[Wa2003] Wacker R.: *Automatische Generierung von Simulationsmodellen auf Basis einer Socket-Lösung mit Planungsdatenbank und dem Simulationssystem Quest von Delmia am Beispiel des Montagewerkes 2 in Tuscaloosa/USA des Unternehmens DaimlerChrysler*. Diplomarbeit, Fachhochschule Esslingen, Institut für Produktionsmanagement und Logistik, 2003.

[We2002] Wenzel S.: Modellbildung in der ereignisdiskreten Simulation. In: Arbeitsgemeinschaft Simulation - ASIM (Hrsg.): ASIM Nachrichten, 2002-2; S. 10–15.

[We2008] Wenzel S., Weiß M., Collisi-Böhmer S., Pitsch, H., Rose O.: *Qualitätskriterien für die Simulation in Produktion und Logistik: Planung und Durchführung von Simulationsstudien*. Springer Verlag, Berlin, 2008.

[We2009] Wenzel S.: *Modellbildung und Simulation in Produktion und Logistik – Stand und Perspektiven*. Dresden: ASIM Workshop, 2009.

[We2012] Weigert G.: *Begriffe & Kennzahlen in Produktion und Logistik*. http://iai8252.inf.tu-dresden.de/twiki/bin/view/BegriffeUndKennzahlen/WebHome. Abgerufen am 29.11.2012.

[WH2007] Wilde T, Hess T.: *Forschungsmethoden der Wirtschaftsinformatik - Eine empirische Untersuchung*. In: WIRTSCHAFTSINFORMATIK 49 Nr. 4, 2007, S.280–287 .

[Wi1999] Wiedemann T.: *Database Oriented Modeling with Simulation Microfunctions*. In: Farrington, P. A.; Nembhard, H. B.; Sturrock, D. T.; Evans, G. W. (Hrsg.): Proceedings of the 1999 Winter Simulation Conference. December 5-8, 1999. Phoenix, AZ, USA, S. 586–590.

[Wi2007] Witte H.: *Allgemeine Betriebswirtschaftslehre: Lebensphasen des Unternehmens und betriebliche Funktionen*. Oldenbourg Wissenschaftsverlag, München, 2007.

[WIKI2012] Wikipedia: *Simulation*. Abruf am 02.02.2012, http://de.wikipedia.org/wiki/Simulation.

[WIKI2012a] Wikipedia: *Vorgehensmodell*. Abruf am 03.02.2012, http://de.wikipedia.org/wiki/Vorgehensmodell.

[WIKI2012b] Wikipedia: *Pattern*. Abruf am 07.08.2012, http://de.wikipedia.org/wiki/Entwurfsmuster.

[WIKI2013a] Wikipedia: *Emulation*. Abruf am 14.02.2013, http://de.wikipedia.org/wiki/Emulation.

[WIKI2013b] Wikipedia: *Warteschlangentheorie*. Abruf am 04.03.2013, http://de.wikipedia.org/wiki/Warteschlangentheorie.

[WIKI2013c] Wikipedia: *Demontage*. Abruf am 15.03.2013, http://de.wikipedia.org/wiki/Zerlegen.

[WIKI2013d] Wikipedia: *Montage*. Abruf am 19.03.2013, http://de.wikipedia.org/wiki/Montage_(Produktion).

[WIKI2013e] Wikipedia: *Quellcode*. Abruf am 24.04.2013, http://de.wikipedia.org/wiki/Quellcode.

[WIKI2013f] Wikipedia: *Extensible Stylesheet Language (XLS)*. Abruf am 29.04.2013, http://de.wikipedia.org/wiki/Extensible_Stylesheet_Language.

[WIKI2013g] Wikipedia: *LINQ*. Abruf am 03.05.2013, http://de.wikipedia.org/wiki/LINQ.

[WIKI2013h] Wikipedia: *Box-Whisker-Plot*. Abruf am 06.05.2013, http://de.wikipedia.org/wiki/Boxplot.

[WIKI2013i] Wikipedia: *Usability/ Benutzerfreundlichkeit*. Abruf am 22.05.2013, http://de.wikipedia.org/wiki/Benutzerfreundlichkeit.

[WIKI2013j] Wikipedia: *Assistent (Wizzard)*. Abruf am 03.06.2013, http://de.wikipedia.org/wiki/Assistent_(Datenverarbeitung).

[WKWI2011] Wissenschaftliche Kommission Wirtschaftsinformatik, Fachbereich Wirtschaftsinformatik (FB WI) in der Gesellschaft für Informatik e.V. (GI): *Profil der Wirtschaftsinformatik, 2011*. Abruf am 01.03.2013, http://www.wirtschaftsinformatik.de//binary/Profil_WI.pdf.

[Wu2000] Wuttke C. C.: *Mehrfachnutzung von Simulationsmodellen in der Produktionslogistik*. Bley H., Hirt G., Weber C. (Hrsg.): Schriftenreihe Produktionstechnik - Band 20, Universität des Saarlandes Produktionstechnik, Saarbrücken, 2000.

[WW2008] Wurdig T., Wacker R.: *Generische Simulationslösung für Fördertechnik*. Rabe M.(Hrsg): Advances in Simulation for Production and Logistics Applications, Fraunhofer IRB Verlag, Stuttgrad, 2008, S. 11-20.

[W3C2012] World Wide Web Consortium (W3C): *Extensible Markup Language (XML) 1.1*. Abruf am 4.01.2012. http://www.w3.org/TR/2006/REC-xml-names11-20060816/.

[W3C2013a] World Wide Web Consortium: *Extensible Stylesheet Language (XSL) Version 1.* Abruf am 29.04.2013. http://www.w3.org/TR/xsl/.

[W3C2013b] World Wide Web Consortium: *XSL Transformations (XSLT) Version 2.0*. Abruf am 29.04.2013. http://www.w3.org/TR/xslt20/.

[W3C2013c] World Wide Web Consortium: *XML Path Language (XPath) 2.0 (Second Edition)*. Abruf am 29.04.2013. http://www.w3.org/TR/ xpath20/4.

[XA2013] Extensible Markup Language For Artificial Neural Networks <XMLANN/>. Abruf am 19.02.2013, http://www.xmlann.org/.

[Ze1997] Zell A.: *Simulation Neuronaler Netze*. Oldenbourg Verlag, Oldenburg, 1997.

[Ze2006] Zenner C.: *Durchgängiges Variantenmanagement in der Technischen Produktionsplanung*. Bley H., Weber C. (Hrsg.) : Schriftenreihe Produktionstechnik - Band 37, Universität des Saarlandes Produktionstechnik, Saarbrücken, 2006.

Betreute Diplom- Master- und Bachelorarbeiten:

[Ba2011] Bambl S.: *Neuronale Netze zur Approximation von Steuerstrategien in Simulationsmodellen der Produktion*. Masterarbeit, TU Ilmenau, Ilmenau, 2011.

[Fi2010] Fiedler A.: *Automatisierte Generierung von Simulationsmodellen unter Verwendung des Core Manufacturing Simulation Data (CMSD) Information Model – Implementierung eines Pilotszenarios*. Diplomarbeit, TU Ilmenau, Ilmenau, 2010.

[Hä2012] Häfer M.: *Konzept für die Datenqualifikation zur simulationsbasierten Optimierung der Produktionsplanung und -steuerung eines Kleinen und Mittelständischen Unternehmens auf Basis des CMSD-Information Model*. Masterarbeit, TU Ilmenau, Ilmenau, 2012.

[Ho2011] Hofmann E.: *Konzeption und Implementierung eines CMSD-Generators als Webanwendung*. Bachelorarbeit, TU Ilmenau, Ilmenau, 2011.

[Ho2012] Hofmann E.: *Auswertungsmöglichkeiten von Simulationsexperimenten durch logistische Kennzahlen auf Basis von CMSD-Daten – Konzeption und Implementierung eines Web-Statistik-Monitors für Simulationsstudien basierend auf CMSD als Datenaustauschformat*. Masterarbeit, TU Ilmenau, Ilmenau, 2012.

[Kl2010] Klemm R.: *Ansätze zur automatischen Modellgenerierung - Ein aktueller Überblick*. Diplomarbeit, TU Ilmenau, Ilmenau 2010.

[Wü2012] Wüstemann, S.: *Automatisierte Generierung von Simulationsmodellen in SLX unter Verwendung des Core Manufacturing Simulation Data (CMSD) Information Model*. Hauptseminar, TU Ilmenau, Ilmenau, 2013.

[Wü2013] Wüstemann, S.: *CMSD-basierte automatische Generierung und Simulation von SLX-Modellen auf Basis von Daten des SAP ERP*. Masterarbeit, TU Ilmenau, Ilmenau, 2013.

Anhang A - Erweiterungen des CMSD-Standards / Liste der Properties

Im Folgenden ist eine Auflistung aller zusätzlich definierten Properties sowie deren Typ und Wertebereich tabellarisch aufgeführt.

Property Bezeichnung	Entität	Datentyp / erlaubte Werte	Kurzbeschreibung
capacity	Ressource {ResourceType= other; nur Puffer}	Integer (<0 entspricht unendlich)	Kapazität eines Puffers
decisionRule	Ressource {ResourceType= other; nur Puffer}	Wert des Aufzählungsdatentyps "DecisionRules"	Reihenfolgeregel, die an einem Puffer Ausgang zum Einsatz kommt
routingRule	Ressource {ResourceType= other; nur Puffer}	Wert des Aufzählungsdatentyps "RoutingRules	Routingregel, die an einem Puffer Ausgang zum Einsatz kommt
availability	Ressource {ResourceType= station OR machine}	Decimal (>=0 ; <=100)	Verfügbarkeit einer Bearbeitungsstation in %
MTTR	Ressource {ResourceType= station OR machine}	Decimal (>=0)	MTTR einer Bearbeitungsstation
reliability	Ressource {ResourceType= station OR machine}	Decimal (>=0 ; <=100)	Gutteilquote einer Bearbeitungsstation in %
setupSkill	SetupDefinition	SkillReference	Referenz auf eine Fähigkeit, die speziell zum Rüsten benötigt wird
repairSkill	Ressource {ResourceType= station OR machine	SkillReference	Referenz auf eine Fähigkeit, die zur Störungsbeseitigung benötigt wird

Anhang B - CMSD Pflichtfelder für die Modellgenerierung und -initialisierung

Im Folgenden ist eine Auflistung aller Pflichtattribute einzelner Entitäten tabellarisch aufgeführt.

CMSD Entität	Attribute				Level*
HeaderSection	CreationTime				4
CMSDDocument Reference	LocalDocumentIdentifier				5
	DocumentLocation				5
Calendar	Identifier				5
	EffectiveStartDate				3
	EffectiveEndDate				3
	Shift	Identifier			5
		StartTime			5
		ApplicableDay			
		Duration	Unit		5
			Value		5
		Break	StartTime		5
			Duration	Unit	5
				Value	5
Resource	Identifier				5
	ResourceType				5
	ResourceClass (ResourceType <> employee)				4
	Name (ResourceType <> employee)				4
	Name (ResourceType =employee)				3
	CurrentStatus				3
	CurrentSetup (ResourceType <> employee; ResourceClass <> puffer)				4

Resource	ShiftAssignment (ResourceClass <> puffer)	CalendarIdentifier			4
		ShiftIdentifier (ResourceType = employee)			4
	GroupDefinitio (ResourceType <> employee)n	ResourceGroupMember	ResourceIdentifier		5
		Connection	Identifier		5
			FromResource	ResourceIdentifier	4
			ToResource	ResourceIdentifier	4
	Property (ResourceType <> employee; ResourceClass <> puffer)	availability	Name		4
			Value		4
		MTTR	Name		*4*
			Value		*4*
		reliability	Name		4
			Value		4
		repairSkill	Name		3
			SkillDefinitionIdentifier		3
			SkillLevelIdentifier		3

	Property (ResourceType <> employee; ResourceClass = puffer)	capacity	Name	4
			Value	4
		decisionRule	Name	4
			Value	4
		routingRule	Name	3
			Value	3
	EmployeeSkill (ResourceType = employee)	SkillDefinitionIdentifier		3
		SkillLevelIdentifier		3
Setup Definition	Identifier			5
	SetupResource	ResourceIdentifier		4
	Property	setupSkill	Name	3
			SkillDefinitionIdentifier	3
			SkillLevelIdentifier	3

Setup ChangeOver	Identifier					5
	CurrentSetup	SetupDefinitionIdentifier				4
	NewSetup	SetupDefinitionIdentifier				4
		ChangoverTime	Unit			4
			Value			4
SkillDefinition	Identifier					5
	SkillLevel			Identifier		5
Part	Identifier					5
PartType	Identifier					5
ProcessPlan	Identifier					5
	PartsProduced		PartType		PartTypeIdentifier	4
	FirstProcess		ProcessIdentifier			3
	Process	Identifier				5
		PartsProduced		PartType	PartTypeIdentifier	3
		PartsConsumed		PartType	PartTypeIdentifier	3
		ResourcesRequired	Resource		ResourceIdentifier	4
			AllowableSetup		SetupDefinition Identifier	4
			Required EmployeeSkill	SkillDefinitionIdentifier		3
				SkillLevelIdentifier		3

		OperationTime	Unit		4
			Value		4
Job	Identifier				5
	Status				4
	PlannedEffort	ReleaseDate			3
		DueDate			3
		ProcessPlan		ProcessPlanIdentifier	4
	ActualEffort	ReleaseDate			3
		DueDate			3
		ProcessPlan		ProcessPlanIdentifier	4
		CurrentProcessPlanStep		ProcessPlanIdentifier	3
				ProcessIdentifier	3
		Event		SequenceNumber	3
				Name	3
				Timestamp	3
Layout	Identifier				5
	AssociatedResource			ResourceIdentifier	4
	Boundary	Width			4
		Depth			4
	Placement	LayoutElementIdentifier			5
		Location	x		4
			y		4

* 5 (Orange) alle im Standard definierten Pflichtfelder;
4 (Gelb) alle für die Modellgeneratoren aktuell zwingend benötigten Attribute;
3 (Khaki) alle für die Modellgenerierung empfohlenen Attribute; gekennzeichnet

Anhang C - Die wichtigsten CMSD Aufzählungsdatentypen

Im Folgenden werden eine Auswahl für diese Arbeit relevanter Aufzählungsdatentypen (enumeration) des CMSD Standards wiedergegeben sowie ggf. im Standard fehlende Werte ergänzt.

Aufzählungsdatentypen (enumeration)	Werte
ResourceStatus [SISO2012a, S. 56]	busy
	idle
	broken
	underMaintenance
	unknown
	setup*
	paused*
DecisionRules [eigner Datentyp, erweiterbar!]	FIFO
	LIFO
	KOZ
	LOZ
	SST
	HCM
	Slack
	Random
RoutingRule [eigner Datentyp, erweiterbar!]	SST
	roundRobin (Reihum)
	Random
JobStatus [SISO2012a, S. 56]]	released
	started
	unknown
	complete
	cancelled
	blocked

Ereignistypen [eigner Datentyp, erweiterbar!]	released
	complete
	start work
	finish work
	start setup
	broken
	Repaired
	start transportation
	finish transportation

* nach Autorenmeinung sinnvolle Erweiterungen, nicht im Standard enthalten

Anhang D - Vollständige Abbildung der Anwendungsfälle zur CMSD Interpretation

Im Folgenden ist eine tabellarische Beschreibung der Basisanwendungsfälle aufgeführt.

(Basis-)Parameter der Bearbeitungsstationen

Parameter	Bearbeitungs-station A	Bearbeitungs-station B	Bearbeitungs-station C	Bearbeitungs-station A1 **
Eingangspuffer-kapazität	5	3	5	5
Ausgangspuffer-kapazität	6	1	1	6
X - Koordinate (Ein; Ausgangspuffer)	250 (200; 300)	300 (250; 350)	350 (300; 400)	250 (200; 300)
Y - Koordinate (Ein; Ausgangspuffer)	100 (100; 100)	200 (200; 200)	300 (300; 300)	100 (100; 100)
*Reparaturfähigkeit**	Skill_ Repair_M1	Skill_ Repair_M2	Skill_ Repair_M2	Skill_ Repair_M1

* Ergänzung im Rahmen des Anwendungsfalls 3;

** Ergänzung im Rahmen des Anwendungsfalls 4

Rüstmatrix Bearbeitungsstation A (und A1*):

Von \ Nach (in Minuten)	ungerüstet	Rüstzustand (RZ) 1	Rüstzustand (RZ) 2	Rüstzustand (RZ) 3
ungerüstet	-	5:00	5:00	5:00
Rüstzustand (RZ) 1	5:00	-	5:20	4:50
Rüstzustand (RZ) 2	5:00	5:20	-	4:50
Rüstzustand (RZ) 3	5:00	3:00	5:50	-

* Ergänzung im Rahmen des Anwendungsfalls 4

Rüstmatrix Bearbeitungsstation B:

Von \ Nach (in Minuten)	ungerüstet	Rüstzustand (RZ) 1	Rüstzustand (RZ) 2	Rüstzustand (RZ) 3
ungerüstet	-	10:00	10:00	10:00
Rüstzustand (RZ) 1	5:00	-	15:00	15:00
Rüstzustand (RZ) 2	5:00	15:00	-	15:00
Rüstzustand (RZ) 3	5:00	15:00	15:00	-

Rüstmatrix Bearbeitungsstation C:

Von \ Nach (in Minuten)	ungerüstet	Rüstzustand (RZ) 1	Rüstzustand (RZ) 2	Rüstzustand (RZ) 3
ungerüstet	-	10:00	10:00	10:00
Rüstzustand (RZ) 1	5:00	-	15:00	15:00
Rüstzustand (RZ) 2	5:00	15:00	-	15:00
Rüstzustand (RZ) 3	5:00	15:00	15:00	-

Arbeitspläne:

Produkt	Nummer des Prozess-schritts	Bearbeit-ungs-station	Benötigter Rüst-zustand	Operationszeit (in Minuten)		Benötigte Werker-fähigkeiten	
				Deterministisch	Stochastisch *	Be-arbeiten	Rüsten
Pro-dukt A	1	A	RZ 1	10:00	Gleichverteilt von 9:00 bis 11:00	Skill_A1 (Level 1)	
	2	B	RZ 1	13:40	-	Skill_A2 (Level 1); Skill_A3 (Level 1)	
	3	C	RZ 1	11:20	-	Skill_B1 (Level 1)	

Pro-dukt B	1	B	RZ 2	20:00	-	Skill_A1 (Level 2)	
						Skill_ Setup_ M2	keiner
	2	C	RZ 2	15:10	-	Skill_A1 (Level 1); Skill_A3 (Level 1)	
	3	A	RZ 1	12:00	Gleichverteilt von 11:30 bis 12:30	Skill_B1 (Level 2)	
Pro-dukt C	1	C	RZ 3	13:30	-	Skill_B2 (Level 1)	
	2	A	RZ 1	10:45	Normalverteilt; Mittelwert: 10:45; Varianz: 0:45	Skill_B1 (Level 2); Skill_B2 (Level 2)	
	3	B	RZ 3	18:00	-	Skill_Setup_M2	Skill_B1 (Level 2)

* Ergänzung im Rahmen des Anwendungsfalls 2

** Ergänzung im Rahmen des Anwendungsfalls 3

Produktionsprogramm (Auszug)

ID des Fertigungsauftrags	Herzustellendes Produkt	Gewünschter Liefertermin	Frühester Freigabetermin
Job_PP1_1	Produkt A	01.02.2014	02.01.2014
Job_PP2_1	Produkt B	03.01.2014	02.01.2014
Job_PP3_1	Produkt C	03.01.2014	02.01.2014
Job_PP3_2	Produkt C	06.02.2014	16.01.2014

Auszug aus einer Kalenderdarstellung in CMSD XML

```
<Calendar>
        <Identifier>Calender_2012</Identifier>
        <Description>Kalender 2012</Description>
        <EffectiveStartDate>2012-01-02T00:00:00</EffectiveStartDate>
        <EffectiveEndDate>2012-12-28T00:00:00</EffectiveEndDate>
        <Shift>
                <Identifier>SchichtFrueh_Werktags</Identifier>
                <Description>Fruehschicht, Werktags</Description>
                <StartTime>05:00:00</StartTime>
                <Duration>
                        <Unit>hour</Unit>
                        <Value>8</Value>
                </Duration>
                <ApplicableDay>monday</ApplicableDay>
                <ApplicableDay>tuesday</ApplicableDay>
                [...]
                <Break>
                        <Description>SchichtFrueh_Werktags_Pause1</Description>
                        <StartTime>07:00:00</StartTime>
                        <Duration>
                                <Unit>minute</Unit>
                                <Value>15</Value>
                        </Duration>
                </Break>
                <Break>
                [...]
        </Shift>
        <Shift>  ...
</Calendar>
```

Werkerfähigkeiten

Werkername	Schicht	Fähigkeiten *
Worker_A	Frühschicht	Skill_A1 (Level 1 und 2), Skill_A2, Skill_B3, Skill_ Repair_M1, Skill_ Setup _M1
Worker_B	Frühschicht	Skill_A3, Skill_B1 (Level 1 und 2), Skill_Repair_M2, Skill_Setup_M2
Worker_C	Frühschicht	Skill_B2 (Level 1 und 2), Skill_B3, Skill_Repair_M3, Skill_Setup_M3
Worker_D	Spätschicht	Skill_A1 (Level 1 und 2), Skill_A3, Skill_ Repair_M1, Skill_ Setup _M1
Worker_E	Spätschicht	Skill_A2, Skill_B1 (Level 1 und 2), Skill_Repair_M2, Skill_Setup_M2
Worker_F	Spätschicht	Skill_A1, Skill_B2 (Level 1 und 2), Skill_B3, Skill_Repair_M3, Skill_Setup_M3

* wenn kein Level angegeben, wird Level1 angenommen; Skill_Setup_MX = Rüstfähigkeit für Bearbeitungsstation X ; Skill_Repair_MX = Reparaturfähigkeit für Bearbeitungsstation X

empfohlene Ressourcenklassen (RessourceClass)

Identifier	Description	ResourceType	Name
Puffer	Ressourcenklasse für Puffer	other	Puffer
Disassembly	Ressourcenklasse für Demontagestation	station	Disassembly
Assembly	Ressourcenklasse für Montagestation	station	Assembly
Slide_Oven	Ressourcenklasse für Schiebeöfen	station	Slide_Oven
Gathering19	Ressourcenklasse für Chargenstationen (Los=19)	station	Gathering19
Gathering20	Ressourcenklasse für Chargenstationen (Los=20)	station	Gathering20
Gathering27	Ressourcenklasse für Chargenstationen (Los=27)	station	Gathering27
Gathering100	Ressourcenklasse für Chargenstationen (Los=100)	station	Gathering100
Ma_simple	Maschinen(einfach)	machine	Maschinen(einfach)

Anhang E - Vollständige CMSD Daten der Anwendungsfälle zur Interpretation des CMSD Standards

Für alle CMSD XML Dateien siehe elektronischen Anhang (beiliegende CD bzw. https://github.com/SoerenBergmann/CMSD-Interpretation)

Anhang F - Liste der Basiskennzahlen im WebStatMonitor

Im Folgenden ist eine tabellarische Liste aller im WebStatMonitor vordefinierten und zur Erstellung eigener Kennzahlen nutzbaren Basiskennzahlen aufgeführt.

Abkürzung	Name der Kennzahl	Englische Bezeichnung	Typ	Sichtweise
BAZ	Bearbeitungszeit	Processing time	absolute Zahl (Zeitangabe)	Auftrags-orientierte (job-oriented)
BLZ	Belegungszeit	Occupied time	Verhältniszahl	
DLZ	Durchlaufzeit	Cycle time	absolute Zahl (Zeitangabe)	
HNZ	Hauptnutzungszeit	Working time	absolute Zahl (Zeitangabe)	
LZ	Liegezeit	Waiting time	absolute Zahl (Zeitangabe)	
SU	Störungsbedingte Unterbrechung	Down time	absolute Zahl (Zeitangabe)	
TRZ	Tatsächliche Rüstzeit	Actual set-up time	absolute Zahl (Zeitangabe)	
AnzTBF	Anzahl fehlerfreier Perioden	Number of zero-defect periods	absolute Zahl	Ressourcen-orientierte (resource - oriented)
AnzTTR	Anzahl der Fehlerbehebungen	Number of repairs	absolute Zahl	
AM	Ausschussmenge	Number of scrap parts	absolute Zahl	
BAZ	Bearbeitungszeit	Processing time	absolute Zahl (Zeitangabe)	
BLZ	Belegungszeit	Occupied time	absolute Zahl (Zeitangabe)	
BZ	Betriebszeit	Operating time	absolute Zahl (Zeitangabe)	
geplanteSZ	Geplante Stillstandszeit	Planned idle time	absolute Zahl (Zeitangabe)	

GM	Gutmenge	Yield	absolute Zahl	Ressourcen-orientierte (resource - oriented)
HNZ	Hauptnutzungszeit	Working time	absolute Zahl (Zeitangabe)	
PBZ	Planbelegungszeit	Planned occupied time	absolute Zahl (Zeitangabe)	
PEZ	Produktionszeit je Einheit (geplante Produktionszeit für alle Aufträge)	Planned production time for all jobs	absolute Zahl (Zeitangabe)	
PM	Produzierte Menge	Output	absolute Zahl	
SU	Störungsbedingte Unterbrechung	Down time	absolute Zahl (Zeitangabe)	
SZ	Stillstandszeit	Idle time	absolute Zahl (Zeitangabe)	
TRZ	Tatsächliche Rüstzeit	Actual set-up time	absolute Zahl (Zeitangabe)	
TTBF	Summe der Zeiten zwischen Fehlern	Total time between failures	absolute Zahl (Zeitangabe)	
TTTR	Summe der Fehlerbehebungszeiten	Total time to repair	absolute Zahl (Zeitangabe)	